# STATE-SPACE REALIZATION FOR NONLINEAR SYSTEMS

A Thesis
Presented to
The Academic Faculty

By

George Fouad Shoukry

In Partial Fulfillment
Of the requirements for the Degree
Master of Science in Mechanical Engineering

Georgia Institute of Technology

December, 2008

# STATE-SPACE REALIZATION FOR NONLINEAR SYSTEMS

Approved by:

Professor Nader Sadegh
School of Mechanical Engineering
*Georgia Institute of Technology*

Professor Ye-Hwa Chen
School of Mechanical Engineering
*Georgia Institute of Technology*

Professor Xu-Yan Chen
School of Mathematics
*Georgia Institute of Technology*

Date Approved: November 5, 2008

To my parents Fouad and Sonia,

and to my brother Michael

# ACKNOWLEDGEMENTS

I wish express my very deep thanks and appreciation for my parents Fouad and Sonia and for my brother Michael. They made me who I am and shaped my personality with their love, help, support, and encouragement. I also wish to express my gratitude to my uncle, Samir, whos advice and guidance helped me get where I am. I also wish to express my deep appreciation and thanks to my advisor, Dr. Nader Sadegh for his guidance, encouragement, patience, and support and for the countless hours he spent with me discussing the work in this thesis. My thanks also go to Dr. Ye-Hwa Chen and Dr. Xu-Yan Chen for being committed and caring teachers and for critiquing this work and being members on my committee. Last but not least, I wish to thank the great professors at Georgia Tech who shared with me their valuable ideas and taught their classes with the utmost commitment and enthusiasm: Dr. Maria Westdickenberg, Dr. William Singhose, Dr. Wayne Book, Dr. John McCuan, Dr. Ronghua Pan, Dr. Tibor Besedes, Dr. Rehim Kilic, and Dr. Maurizio Iacopetta.

# TABLE OF CONTENTS

# LIST OF FIGURES

# SUMMARY

The state-space realization problem is a very basic and fundamental problem of control theory. The topic is also becoming increasingly important as practitioners of both physical and social sciences find it crucial to model very complex systems based on input-output data only. In this thesis, a review of the topic will be given for general nonlinear systems and for the less general linear case as well. The thesis will also present some new theoretical results that contribute to the development of the state-space realization topic. Specifically, an important result will show that if a system can be identified by an input-output equation of a particular form, which is fairly general, then a state-space realization can always be easily derived directly from the input-output map. Finally, the theory will be applied to find a state-space model for a nonlinear hydraulic system based on its input-output data.

# CHAPTER I

# INTRODUCTION AND LITERATURE REVIEW

At the base of many scientific disciplines is mathematical modeling. The natural human curiosity to understand various processes and events leads mathematical modeling to be an important first step in many natural and social sciences. Usually, the goal of a model is to allow the user to predict the future to a certain degree of accuracy, but this need not be the only use of mathematical models. They are also extensively used in designing new systems and in analyzing existing systems to assess their efficiency and examine possible improvements, among other things. Due to the existence of many different kinds of systems and various modeling techniques, modeling a system is not always trivial. In control theory, the three most basic steps to control a system are modeling, analysis, and finally, control. Modeling can be viewed as the most error-prone step of the three. This is because once a model is chosen, many results and conclusions can be deduced logically, albeit not always trivially, from the model, but the act of modeling requires assumptions to be made and the validity of those assumptions affects all the results obtained based on the model. Not only are the assumptions an issue, but the model chosen, given a certain set of assumptions, may be a worse or a better choice in the given situation than another model based on the same set of assumptions. As a simple example, consider modeling a mechanics problem using Newtonian mechanics versus Lagrangian mechanics. Although the assumptions of one model are also true in the other model (even if they are not explicitly specified), the two models use different ideas and modeling techniques, which in certain cases lead to the use of one of them to be much easier or computationally faster than the other. As an example of where different assumptions lead to different

models, consider modeling projectile motion under the assumption that air drag is negligible versus under the assumption that air drag is significant. An example from the social sciences is the different international trade models (i.e. Ricardian, Specific-Factors, Hecksher-Ohlin) that result from different assumptions about technologies and factors of production. The complexity of the topic of mathematical modeling is immediately apparent when the vast amount of possibilities and techniques are taken into consideration. Hence, the systematic study of mathematical modeling of systems in general is very important.

Due to the extreme generality of the concept of a system, many classifications are commonly used to group similar systems together. For example, one may consider the class of all linear systems, or that of all deterministic systems that have no random variables or random external influences. Due to the large differences arising between different classes of systems, it is necessary to only consider a certain subset of systems such that the analysis remains valid when applied to any system in that subset. The subset of systems dealt with in this thesis have the following characteristics: causal, dynamic, stationary (i.e. time invariant), discrete, and deterministic. Any system considered contains variables, which may or may not be important depending on the specific problem of interest. Out of the different variables in a system, the minimal set of variables needed to completely describe the behavior of the system at any point in time are the *state variables*. The number of the state variables in a dynamic system is the order of the system. Other than the state variables, there are also *inputs* and *outputs*. Inputs summarize all the external effects on the system and can, when possible, also be used to control the system's state, or a part of it, to a certain point in the state space or to force it to follow a specific trajectory. Outputs are usually comprised of one or a few states of the system, those which are measurable by sensors, and can in general be a function of the state variables. Inputs can also be used (when possible) to control the output, which is indirectly just controlling the state variables.

Since the inputs and outputs are external to the system, they will be referred to as *external* variables and since the states are properties of the system, they will be the *internal* variables.

This thesis will examine the relationship between two very general models that are commonly used to model dynamic systems: state-space models and input-output models. Specifically, the thesis will address the topic of state space realization, which is the question of when and how one can go from an input-output model to a state space model. An introduction to the two different models and a brief discussion of the advantages of state-space models over input-output models will be presented in the next few sections.

## *1.1 State-Space Models*

For continuous time systems, state-space models use a system of first order ordinary differential equations to describe the dynamic behavior of the state variables. The model is comprised of an equation for each of the state variables that describes how this variable changes in time as a function of the other state variables and the external inputs. For discrete time systems, difference equations are used instead. The state space model also includes an equation for the output(s) that show how the output(s) relate to the state variables. In this thesis, the only systems considered will be discrete and single input-single output (SISO). Therefore, the discussion will proceed with only those systems in consideration. However, it should be noted that for the purposes of the work in this thesis, the techniques developed for SISO systems can be generalized to multi input-multi output (MIMO) systems. On the other hand, discrete systems' techniques are not usually readily applicable to continuous system models due to the inherent differences (differential vs. difference equations) in the modeling techniques. Despite the limited applicability to continuous time systems, the discrete time methods are used extensively in applications due to the dominance of digital

machines and they also apply more easily to the social sciences where observations can only be made at discrete time intervals that can be as short as seconds or as long as years. Furthermore, it will *not* be assumed that the systems under consideration are linear because linear systems are only a small subset of the much more general class of not-necessarily-linear systems, which will be discussed throughout the thesis. When the term "nonlinear" is used in this thesis, it will not be used to imply strictly nonlinear, but instead, it will be taken to mean not-necessarily-linear.

Following the previous paragraph, a state space model used in this thesis will have the form:

$$\begin{aligned} \mathbf{x}[k+1] &= f\left(\mathbf{x}[k], u[k]\right) \\ y[k] &= h(\mathbf{x}[k]) \end{aligned} \tag{1}$$

where, $\mathbf{x}[k] \in \mathbb{R}^n$ and $u[k] \in \mathbb{R}$ and $y[k] \in \mathbb{R}$ denote the state vector, input, and output respectively, and $f(.,.)$ and $h(.)$ are smooth functions. The value in the brackets following the variables denotes the time step at which the variables are being considered. For example, $\mathbf{x}[k+1]$ denotes the state vector evaluated at the $k+1$'th step. State-space models are not unique and any dynamic system can have an infinite number of state-space models. Given any state-space model for a system, one can redefine the state variables and then rewrite the equations to obtain a different, but equivalent, state-space model [13].

The nature of state-space models make them very desirable for analyzing or designing a system [18]. First, state-space models can very easily and naturally handle nonlinearity in the model. In fact, most of nonlinear systems control theory is based on state space models [13], which in itself is a very good reason to attempt to model a system using state-space because that would increase the applicability of nonlinear systems methods. Second, state-space models give a clear description of how each of the state variables changes, which can be generalized to how the state vector

changes in the state space. In the lower dimension cases, a visual representation of the state trajectory can be constructed, which in some cases can make the intuitive understanding of the system much deeper. Also, the fact that state-space models explicitly express how the internal variables in a given dynamic system change makes them, usually, the easiest starting point to answering some inherent questions about the system such as stability, controllability, and observability. Controllability is the question of whether we can use the external input to force the system state from any point in the state space to any other point in a finite amount of time. Observability is the question of whether, using only input-output data, we can construct the initial state vector at which the system started. Formal definitions for controllability and observability will be given in a later section, but it can be readily seen that answering these questions is very important to dynamic systems' analysis. Finally, state space models can very naturally handle MIMO systems since the basic elements in state-space models are, in general, vectors.

If the system is observable, starting from a state-space model, one can always transform the model into an input-output model. This can be intuitively seen from (1), because the output is a function of the state vector, and the state equations can be used to construct the state trajectory as a function of the input and initial conditions. Therefore, when comparing state-space models with input-output models, one does not lose anything by having a state-space model, because the input-output model can be derived from the state space model when the system is observable. Hence, it may seem that state-space models are always more useful than input-output models, which begs the question of why one would use an input-output model at all. The answer to this question is simply due to the fact that input-output models are usually easier to construct than state-space models. State-space models are usually derived from first principles [6], which implies complete knowledge of the variables in the system and the ability to describe how they change using dynamic equations. This is usually not

the case when analyzing complex systems, especially ones where nonlinearity heavily influences the system. When it is too difficult to construct a state space model, input-output models, which will be discussed in the next section, are often used to describe the system.

## *1.2 Input-Output Models*

An input-output model is simply a relationship between the current output of the system and the previous inputs and outputs at time steps in the past. The order of the system dictates how many time steps in the past affect the current output. When analyzing complex systems where dynamic equations are hard to construct, input-output relationships prevail due to the relative ease at which they can be constructed. Constructing input-output relations for dynamic systems falls under the field of system identification; a comprehensive treatment of the subject can be found in [9].

Constructing input-output models is a two step process: First, a form of the model has to be chosen; and second, a method for estimating the parameters of the model has to be used. Many different forms and methods for estimating the parameters exist, allowing for a realistically endless number of combinations. Two common types of models are non-linear auto-regressive moving average with exogenous input (NARMAX) [6] [7] and feedforward neural network models [8]. A common method for estimating the parameters is ordinary least squares (OLS), which was developed by Gauss [2]. In chapter 4, nonlinear least squares will be used, which uses the same idea to estimate the parameters in models which are nonlinear in parameters [19].

Input-output models are sufficient when one is simply trying to get an input/output relationship. However, since they model only the external variables, the operation of the internal variables is not very clear from an input-output model. Therefore, they are not suitable for many techniques of dynamic system analysis and design. For

those purposes, and especially for the purpose of control, state-space models are a better candidate to use. The combination of the ease at which input-output models can be constructed and the versatility of state-space models makes the topic of state-space realization, the subject of the next section, a very fundamental and important topic in control theory.

## *1.3 State-Space Realization*

State-space realization, or the study of transforming input-output models into state-space models, is fundamentally important for control theory. Over the past two and a half decades, much research has been done in that area. For linear systems, state-space realization theory is well established and a good overview can be found in [15]. For nonlinear systems, however, the topic is much more involved. The early results for continuous systems are given in [16] and [18] and for discrete systems in [14], [3], [10], [6], [11].

For continuous systems, [16] presented necessary and sufficient conditions for replacement of a higher order implicit differential equation (input-output map) to a set of first order explicit differential equations (state-space model) and gave a constructive procedure for obtaining the realization. Some differential geometric methods were used; an introduction to differential geometry can be found in [1], and a brief introduction to differential geometry as it relates to nonlinear systems is given in [17].

For discrete systems, [14] used algebraic geometry and commutative algebra to study the realization of polynomial input-output equations and [3] studied invertible realization of nonlinear systems. In [10] and [11], Monaco and Norman-Cyrot studied the realization of input-output maps expressed as volterra series.

More recently, Sadegh [12] and Kotta [4] used two different approaches to find necessary and sufficient conditions for a general I/O map to be realizable in state space form and in [5] it was shown that both approaches were equivalent. Sadegh [12]

developed a general class of I/O maps that satisfied the necessary and sufficient conditions for realizability, and therefore were guaranteed to have a state space realization, which could be easily found using an algorithm presented in [12].

## *1.4 Problem Statement*

The purpose of this research is to study the realization problem for discrete time, nonlinear systems and present some new theoretical results. The theory will then be applied to a simple system to illustrate how it can be used. First, the thesis will address the necessary and sufficient conditions for an input-output map to be realizable in state-space form. It will be shown that a simplification to the necessary and sufficient conditions exists. Second, the thesis will present a general form for the input-output map, which is guaranteed to satisfy the necessary and sufficient conditions. The form presented in the thesis is more general than the ones found currently in the literature.

## *1.5 Thesis Organization*

The following chapters will present a thorough explanation of state-space realization theory starting from the basics, followed by the presentation of some new results that are not in the literature, and finally an application will be used to show how the theory can be applied.

In chapter 2, the main results of Sadegh [12] will be presented in detail along with their proofs. A firm understanding of these results is crucial to understand the following chapters. The results in this chapter are concerned with the realization problem for nonlinear systems and the methods developed can also be used for linear systems due to their generality. The last section of chapter 2 will show how the results can be applied to the less general linear case.

Chapter 3 will introduce some novel theoretical results and will serve as the main contribution of the thesis. The results in this chapter build on the work of Sadegh

presented in the chapter 2 and use it as a basis to develop some new results.

Chapter 4 applies the theory developed in chapter 3 to find a state-space model for hydraulic system starting from input-output data.

Finally, chapter 5 makes some concluding remarks and discusses the possibilities of further work.

# CHAPTER II

# STATE-SPACE REALIZATION

## *2.1 Chapter Outline*

This chapter will reintroduce the derivations and the results from paper [12], because an understanding of them is crucial to understanding the results of the next chapter, which will mainly build on the ones introduced here. It should be stressed that the work in this chapter was done by Sadegh and can be found in [12].

At first, some preliminaries and definitions will be presented. The problem will be formalized and some assumptions will be stated. Then, the block state-space representation of systems will be presented, which will notably simplify the following derivations and proofs. Once the preliminaries are presented, the concept of Dynamical Input-Output Maps (DIOM's) will be presented, and it will be shown how this concept is essential in determining whether an input-output map has a state-space realization or not. In fact, it will be shown that an input-output map has a state-space realization if and only if it is DIOM, thus giving necessary and sufficient conditions for existence of state-space realizations. Next, an equivalent formulation of the necessary and sufficient conditions will be given to simplify the computation needed. Finally, two algorithms will be presented: one for checking whether an input-output map has a state-space realization, and one for finding the state-space realization when possible. Some examples will also be given to show how the previous concepts can be applied.

## *2.2 Problem Statement*

Here, the problem will be formulated to set the stage for the following sections. As mentioned in the introduction, we assume single input-single output systems for

simplicity. Consider a general, discrete, input-output (I/O) map given by

$$y[k] = g(y[k-m], \ldots, y[k-1], u[k-m], \ldots, u[k-1]) \tag{2}$$

where, $u[k] \in \mathbb{R}$ and $y[k] \in \mathbb{R}$ represent the input and output of the process, respectively, $g$ is a smooth function, and $m$ is the order of the system. One assumption will be made here that will used throughout the rest of the thesis. The assumption is that there exist constant equilibrium input and output $\bar{u}$ and $\bar{y}$, respectively so that $\bar{y} = g(\bar{y}, \ldots, \bar{y}, \bar{u}, \ldots, \bar{u})$. Furthermore, without loss of generality, it will be assumed that $\bar{u} = \bar{y} = 0$. This assumption can be made without loss of generality, because for example, if the equilibrium was at $\bar{y} = 0$ and some $\tilde{u} = c \neq 0$. Then a new input, $\bar{u}$, can be defined such that $\bar{u} := \tilde{u} - c$, which implies that the new input equilibrium is at $\bar{u} = 0$. Similarly, the output equilibrium can always be redefined to be equal to 0.

Now, the problem can be introduced. The main problem is to find necessary and sufficient conditions for the existence of a state-space realization of the form given in 1. The state-space realization must be observable, which will be formally defined in a later section. The next few sections will address the state-space realization problem in detail.

## *2.3 State Space Realization Preliminaries*

The goal of this section is to formally define the concepts of controllability and observability as they were introduced in [12], which will be central to the state-space realization problem. Before the definitions are introduced, two assumptions will be presented that will be used throughout the rest of the chapter. The first assumption is that the system given by (1) is invertible, so that the Jacobian of $f$ with respect to $\mathbf{x}$ is nonsingular everywhere. The second assumption is that there exist constant equilibrium input $\bar{u}$ and state $\bar{x}$ such that $\bar{x} = f(\bar{x}, \bar{u})$, and without loss of generality, it will be assumed that $\bar{u} = \bar{x} = 0$.

A convenient way to represent systems is through the block state-space representation, which will help with the definitions of controllability and observability and will be used in the derivations in this chapter. To start, we define the block of $m$ inputs and outputs by

$$\mathbf{u}[k] = (u_1[k], \ldots, u_m[k]), \; u_i[k] := u[k+i-1] \text{ and} \tag{3}$$

$$\mathbf{y}[k] = (y_1[k], \ldots, y_m[k]), \; y_i[k] := y[k+i-1] \tag{4}$$

respectively. Then, the state transition map of the system starting from an initial state $\mathbf{x}$ and input sequence $u_1, u_2, \ldots, u_i$ will be defined. The state transition map is the value of the states at time step $i+1$ after applying the inputs $u_1, u_2, \ldots, u_i$. The state transition map will be defined by

$$f^i(\mathbf{x}, \mathbf{u}) := f_{u_i} \circ f_{u_{i-1}} \circ \cdots \circ f_{u_2} \circ f_{u_1}(\mathbf{x})$$

Applying the state (1) to evaluate $\mathbf{x}[k+1], \ldots, \mathbf{x}[k+m]$, we get

$$\begin{aligned} \mathbf{x}[k+m] &= f^m(\mathbf{x}[k], \mathbf{u}[k]) \\ \mathbf{y}[k] &= h^m(\mathbf{x}[k], \mathbf{u}[k]) \end{aligned} \tag{5}$$

where, $h^i(\mathbf{x}, \mathbf{u}) = (h(\mathbf{x}), h \circ f^1_{\mathbf{u}}(\mathbf{x}), \ldots, h \circ f^{i-1}_{\mathbf{u}}(\mathbf{x}))$. Since the definitions of controllability and observability require the usage of partial derivatives of the maps of $f^m$ and $h^m$, some differential geometry tools will be introduced to achieve that purpose. For a smooth diffeomorphism ("flow") $\phi(\mathbf{x}) \in \mathbb{R}^n$, vector-field $\mathbf{v}(\mathbf{x}) \in \mathbb{R}^n$, and covector field $\mathbf{w}(\mathbf{x}) \in \mathbb{R}^{1\times n}$, define the $\mathrm{Ad}_\phi$ and $\mathrm{Ad}^*{}_\phi$ operators as follows

$$\begin{aligned} \mathrm{Ad}_\phi \mathbf{v}(\mathbf{x}) &: = [D\phi(\mathbf{x})]^{-1} \mathbf{v} \circ \phi(\mathbf{x}) \\ \mathrm{Ad}^*{}_\phi \mathbf{w}(\mathbf{x}) &: = [\mathbf{w} \circ \phi(\mathbf{x})] D\phi(\mathbf{x}) \end{aligned}$$

Using those definitions with the chain rule, it can be seen that

$$\begin{aligned} \mathrm{Ad}_{\phi_i \circ \cdots \circ \phi_1} \mathbf{v} &= \mathrm{Ad}_{\phi_1} \cdots \mathrm{Ad}_{\phi_i} \mathbf{v} \\ \mathrm{Ad}^*{}_{\phi_i \circ \cdots \circ \phi_1} \mathbf{w} &= \mathrm{Ad}^*{}_{\phi_1} \cdots \mathrm{Ad}^*{}_{\phi_i} \mathbf{w} \end{aligned}$$

The definitions of $\mathrm{Ad}_\phi$ and $\mathrm{Ad}^*{}_\phi$ can also be extended to include matrices of vectors and covectors. For example, for smooth diffeomorphisms $\phi_1, \ldots, \phi_i : \mathbb{R}^n \to \mathbb{R}^n$,

$$\mathrm{Ad}_\phi [\mathbf{v}_1, \ldots, \mathbf{v}_n] = [\mathrm{Ad}_\phi \mathbf{v}_1, \ldots, \mathrm{Ad}_\phi \mathbf{v}_n] \text{ and}$$
$$\mathrm{Ad}^*{}_\phi (\mathbf{w}_1, \ldots, \mathbf{w}_n) = (\mathrm{Ad}^*{}_\phi \mathbf{w}_1, \ldots, \mathrm{Ad}^*{}_\phi \mathbf{w}_n)$$

Now, we define the vector and covector fields

$$B_u(\mathbf{x}) := [D_\mathbf{x} f(\mathbf{x}, u)]^{-1} D_u f(\mathbf{x}, u),\ C(\mathbf{x}) := D_\mathbf{x} h(\mathbf{x})$$

which allows us to find a compact representation of the partial derivatives of the $f^m$ and $h^m$ maps. The partial derivatives of functions $f^m$ and $h^m$ are given by

$$\begin{aligned} D_\mathbf{x} f^m(\mathbf{x}, \mathbf{u}) &= \mathrm{Ad}^*{}_{f_{u_1}} \cdots \mathrm{Ad}^*{}_{f_{u_m}} I_{n\times n}(\mathbf{x}) \\ D_\mathbf{u} f^m &= D_\mathbf{x} f^m [B_1, \ldots, B_m] \end{aligned}$$

$$D_\mathbf{x} h^m = \begin{bmatrix} C_0 \\ C_1 \\ \vdots \\ C_{m-1} \end{bmatrix},\ D_\mathbf{u} h^m = \begin{bmatrix} 0 & 0 & \cdots & 0 \\ C_1 B_1 & 0 & \cdots & 0 \\ \vdots & \ddots & \ddots & 0 \\ C_{m-1} B_1 & \cdots & C_{m-1} B_{m-1} & 0 \end{bmatrix}$$

where

$$B_i(\mathbf{x}, \mathbf{u}) = \mathrm{Ad}_{f_{u_1}} \cdots \mathrm{Ad}_{f_{u_{i-1}}} B_{u_i}(\mathbf{x})$$
$$C_i(\mathbf{x}, \mathbf{u}) = \mathrm{Ad}^*{}_{f_{u_1}} \cdots \mathrm{Ad}^*{}_{f_{u_i}} C(\mathbf{x})$$

Finally, we can introduce the definitions of controllability and observability.

**Definition 1** *The state-space space representation is said to be* controllable *(at the first order) if* $\mathrm{rank}\,[D_\mathbf{u} f^n(\mathbf{0}, \mathbf{0})] = n$ *. It is said to be* observable *(at the first order) if* $\mathrm{rank}\,[D_\mathbf{x} h^n(\mathbf{0}, \mathbf{0})] = n$*. It is said to be* minimal *(at the first order) if it is both controllable and observable.*

As a remark, the definitions given for controllability and observability are equivalent to requiring controllability and observability of the linear system resulting from linearizing (1) about the origin. Next, the importance of controllability and observability will be shown using the following proposition which formalizes the informal definitions for controllability and observability given in the chapter 1.

**Proposition 2**

*i) If the system given by (1) is controllable, then there exists an open neighborhood* $\mathcal{X} \subset \mathbb{R}^n$ *of the origin and a smooth (unique if* $m = n$*) function* $\psi : \mathcal{X} \times \mathcal{X} \to \mathbb{R}^m$*, for all* $m \geq n$*, such that if* $\mathbf{u} = \psi(\mathbf{x}_m, \mathbf{x}_0)$ *then* $\mathbf{x}_m = f^m(\mathbf{x}_0, \mathbf{u})$*,* $\forall \mathbf{x}_0, \mathbf{x}_m \in \mathcal{X}$*.*

*ii) If the system given by (1) is observable, then there exist open neighborhoods* $\mathcal{X} \subset \mathbb{R}^n$*,* $\mathcal{U} \subset \mathbb{R}$*,* $\mathcal{Y} \subset \mathbb{R}$ *of the origin and a smooth (unique if* $m = n$*) function* $\Omega : \mathcal{Y}^m \times \mathcal{U}^m \to \mathcal{X}$*, for all* $m \geq n$*, such that if* $\mathbf{y} = h^m(\mathbf{x}, \mathbf{u})$ *then* $\mathbf{x} = \Omega(\mathbf{y}, \mathbf{u})$*,* $\forall \mathbf{x} \in \mathcal{X}$ *and* $\mathbf{u} \in \mathcal{U}^m$*.*

## *2.4 Dynamical Input-Output Maps*

As was mentioned in the introduction, if an observable state-space model for a system is available, an I/O map can always be generated. This section will establish the properties of I/O maps that are generated from state-space models. These properties will form the necessary conditions for an I/O map to have a state-space realization, and in the next section it will be shown that they are also sufficient. Before we derive the necessary conditions, the notation used for *advancement* will be introduced. We

will use $\{.\}_p$ to denote advancing by $p$ times. A few examples:

$$\begin{aligned}
\{y_1\}_1 &= y_2 \\
\{u_2\}_2 &= u_4 \\
\{y_m\}_1 &= y_{m+1} \\
&= g(y_1, \ldots, y_m, u_1, \ldots, u_m) \\
\{y_m\}_2 &= g(y_2, \ldots, y_m, g(y_1, \ldots, y_m, u_1, \ldots, u_m), u_2, \ldots, u_{m+1})
\end{aligned}$$

Also, it will be seen later that any $u_i$ with $i > m$ will be special, so instead of writing those as $u_{m+j}$, we will use $v_j$ instead. For example:

$$\begin{aligned}
\{u_m\}_1 &= v_1 \\
\{u_{m-1}\}_4 &= v_3 \\
\{v_3\}_2 &= v_5 \\
\{y_m\}_2 &= g(y_2, \ldots, y_m, g(y_1, \ldots, y_m, u_1, \ldots, u_m), u_2, \ldots, u_m, v_1)
\end{aligned}$$

Using the notation introduced in (3) and evaluating $y[k]$, $y[k+1]$,..., $y[k+m-1]$, recursively in terms of $y[k-m], \ldots, y[k-1]$ and $u[k-m], \ldots, u[k+m-2]$, we get

$$\mathbf{y}[k] = G(\mathbf{y}[k-m], \mathbf{u}[k-m], \mathbf{u}[k]) \tag{6}$$

where the $i$–th row of $G$, denoted by $g_i$, $i = 1, \ldots, m$, is given by

$$g_i(\mathbf{y}, \mathbf{u}, \mathbf{v}) = g(y_i, \ldots, y_m, g_1(\mathbf{y}, \mathbf{u}, \mathbf{v}), \ldots, g_{i-1}(\mathbf{y}, \mathbf{u}, \mathbf{v}), u_i, \ldots, u_m, v_1, \ldots, v_{i-1})$$

for $i \geq 2$ and $g_1(\mathbf{y}, \mathbf{u}, \mathbf{v}) = g(y_1, \ldots, y_m, u_1, \ldots, u_m)$. For future references we also define

$$g_{m+1}(\mathbf{y}, \mathbf{u}, \mathbf{v}) = g(g_1(\mathbf{y}, \mathbf{u}, \mathbf{v}), \ldots, g_m(\mathbf{y}, \mathbf{u}, \mathbf{v}), v_1, \ldots, v_m)$$

Now, properties of an I/O map that is derived from an observable state-space realization will be examined. Due to the assumption that the system is observable and using proposition (2), $\mathbf{x}[k]$ can be solved for locally in terms of $\mathbf{y}[k]$ and $\mathbf{u}[k]$. So

that implies that $\mathbf{x}[k] = \Omega(\mathbf{y}[k], \mathbf{u}[k])$ for some local function $\Omega(.,.)$. Using this and substituting for $\mathbf{x}[k]$ in (5) yields

$$\mathbf{x}[k+m] = \Theta(\mathbf{y}[k], \mathbf{u}[k]) := f^m(\Omega(\mathbf{y}[k], \mathbf{u}[k]), \mathbf{u}[k])$$

or $\mathbf{x}[k] = \Theta(\mathbf{y}[k-m], \mathbf{u}[k-m])$. Therefore, the input-output map can be written as

$$y[k] = h(\mathbf{x}[k]) = h(\Theta(\mathbf{y}[k-m], \mathbf{u}[k-m])) \tag{7}$$

The corresponding block input–output equation becomes

$$\mathbf{y}[k] = G(\mathbf{y}[k-m], \mathbf{u}[k-m], \mathbf{u}[k]) := h^m(\Theta(\mathbf{y}[k-m], \mathbf{u}[k-m]), \mathbf{u}[k]) \tag{8}$$

Now, the formal definition of a DIOM will be given and the key properties of a DIOM will be introduced.

**Definition 3** *The input–output map* $y[k] = g(y[k-m], \ldots, y[k-1], u[k-m], \ldots, u[k-1])$ *is said to be a Dynamical Input Output Map (DIOM) if it is the unique input–output map (7) corresponding to an* $m$*–dimensional observable state–space realization.*

Finally, the following lemma will introduce some important properties of a DIOM, which will serve as necessary condition for an I/O map to be realizable.

**Lemma 4** *Let* $G(.,.,.)$ *be the block input–output map of a DIOM given by (8). Then* $D_{\mathbf{y}}G(\mathbf{y}, \mathbf{u}, \mathbf{v})$ *is nonsingular and* $[D_{\mathbf{y}}G(\mathbf{y}, \mathbf{u}, \mathbf{v})]^{-1} D_u G(\mathbf{y}, \mathbf{u}, \mathbf{v})$ *is independent of the third variable* $\mathbf{v}$ *on a neighborhood of the origin.*

**Proof.** Let the $m$–th order system

$$\mathbf{x}[k+1] = f(\mathbf{x}[k], u[k]), \;\; y[k] = h(\mathbf{x}[k])$$

be an observable realization of the DIOM. Then the block input–output map of the DIOM is given by $G(\mathbf{y}, \mathbf{u}, \mathbf{v}) = h^m(\Theta(\mathbf{y}, \mathbf{u}), \mathbf{v})$. Differentiating this with respect to $\mathbf{y}$

and $\mathbf{u}$ using chain rule, we get

$$D_{\mathbf{y}}G(\mathbf{y},\mathbf{u},\mathbf{v}) = D_{\mathbf{x}}h^m(\mathbf{x},\mathbf{v})\ D_{\mathbf{y}}\Theta(\mathbf{y},\mathbf{u})$$
$$D_{\mathbf{u}}G(\mathbf{y},\mathbf{u},\mathbf{v}) = D_{\mathbf{x}}h^m(\mathbf{x},\mathbf{v})\ D_{\mathbf{u}}\Theta(\mathbf{y},\mathbf{u})$$

Differentiating $\mathbf{y} = h^m(\Omega(\mathbf{y},\mathbf{u}),\mathbf{u})$, and $\Theta(\mathbf{y},\mathbf{u}) = f^m(\Omega(\mathbf{y},\mathbf{u}),\mathbf{u})$ with respect to $\mathbf{y}$ yields

$$D_{\mathbf{y}}\Theta(\mathbf{y},\mathbf{u}) = D_{\mathbf{x}}f^m(\mathbf{x},\mathbf{u})D_{\mathbf{y}}\Omega(\mathbf{y},\mathbf{u})$$
$$I_{m\times m} = D_{\mathbf{x}}h^m(\mathbf{x},\mathbf{u})D_{\mathbf{y}}\Omega(\mathbf{y},\mathbf{u})$$

By the assumption made in the last section, $D_{\mathbf{x}}f^m(\mathbf{x},\mathbf{u})$ is invertible. This together with the preceding equations imply that $D_{\mathbf{y}}\Theta(\mathbf{y},\mathbf{u})$ is also invertible on a neighborhood of the origin. Thus

$$\left(D_{\mathbf{y}}G(\mathbf{y},\mathbf{u},\mathbf{v})\right)^{-1} D_{\mathbf{u}}G(\mathbf{y},\mathbf{u},\mathbf{v}) = \left(D_{\mathbf{y}}\Theta(\mathbf{y},\mathbf{u})\right)^{-1} D_{\mathbf{u}}\Theta(\mathbf{y},\mathbf{u})$$

is independent of $\mathbf{v}$. ■

## *2.5 State-Space Realization of a DIOM*

In this section, the necessary and sufficient conditions for an I/O map to be realizable will be given. The results in this section are very important, because the next chapter will mainly build on them and introduce new results. In addition, two algorithms, that were also given in [12], will be shown in this section to illustrate how a state-space realization can be found in theory, and finally, an example will be used to illustrate how they can be applied.

First, a suitable state vector will be constructed and then used to prove the necessary and sufficient conditions in order for an I/O map to be DIOM. Given an I/O map (2) with the block I/O map defined by (6), let the state vector candidate be

$$\mathbf{x}[k] = G(\mathbf{y}[k-m],\mathbf{u}[k-m],\mathbf{0}) \tag{9}$$

In order for $\mathbf{x}[k]$ to qualify as a state vector, we need to show that $\mathbf{x}[k+1]$ is a function of $\mathbf{x}[k]$ and $u[k]$ only, and does not depend on $u[j]$, $\forall j < k$. Incrementing $k$ in the preceding equation by 1, it can be seen that

$$\mathbf{x}[k+1] = G(\mathbf{y}[k-m+1], \mathbf{u}[k-m+1], \mathbf{0}) =: G_1(\mathbf{y}[k-m], \mathbf{u}[k-m], \mathbf{v}[k])$$

where

$$\mathbf{v}[k] = (u[k], 0, \ldots, 0)\,, \text{ and } G_1 = (g_2, g_3, \ldots, g_m, g_{m+1}) \tag{10}$$

Since the Jacobian of $G(\mathbf{y}, \mathbf{u}, \mathbf{0})$ with respect to $\mathbf{y}$ is nonsingular, by the implicit function theorem $\mathbf{y}[k-m]$ can be solved for in terms of $\mathbf{x}[k]$, $\mathbf{u}[k-m]$, and $\mathbf{v}[k]$. That is, there exists a smooth function $G_y^{-1}$ such that

$$G(G_y^{-1}(\mathbf{x}, \mathbf{u}, \mathbf{v}), \mathbf{u}, \mathbf{v}) = \mathbf{x} \tag{11}$$

locally. Thus

$$\mathbf{x}[k+1] = G_1(G_y^{-1}(\mathbf{x}[k], \mathbf{u}[k-m], \mathbf{0}), \mathbf{u}[k-m], \mathbf{v}[k]) \tag{12}$$

The following lemma is very important, because it shows that if we have an I/O map that satisfies the conditions in Lemma (4), then (12) is an observable state-space realization for that I/O map.

**Lemma 5** *Suppose that the block I/O map (6) is such that $D_{\mathbf{y}}G(\mathbf{y}, \mathbf{u}, \mathbf{v})$ is nonsingular and that*

$$[D_{\mathbf{y}}G(\mathbf{y}, \mathbf{u}, \mathbf{v}]^{-1}[D_{\mathbf{u}}G(\mathbf{y}, \mathbf{u}, \mathbf{v})]$$

*is independent of the third variable $\mathbf{v}$ on a neighborhood of the origin. Defining the state vector $\mathbf{x}[k] = G(\mathbf{y}[k], \mathbf{u}[k], \mathbf{0}) \in R^m$, then*

$$\begin{aligned} \mathbf{x}[k+1] &= f\left(\mathbf{x}[k], u[k]\right) := G_1(G_y^{-1}(\mathbf{x}[k], \mathbf{0}, \mathbf{0}), \mathbf{0}, \mathbf{v}[k]) \\ y[k] &= h\left(\mathbf{x}[k]\right) := x_1[k] \end{aligned} \tag{13}$$

*is an $m$–dimensional observable state–space realization for the I/O map.*

**Proof.** First, to show that (12) is a valid state-space realization, we need to show that the right hand side does not depend on $\mathbf{u}[k]$. Therefore, if we use the chain rule and take the derivative of both sides of (13), we need to show that

$$D_{\mathbf{u}}G_1(G_y^{-1}(\mathbf{x},\mathbf{u},0),\mathbf{u},\mathbf{v}) = \begin{pmatrix} D_{\mathbf{y}}G_1(\mathbf{y},\mathbf{u},\mathbf{v})D_{\mathbf{u}}G_y^{-1}(\mathbf{x},\mathbf{u},0) \\ +D_{\mathbf{u}}G_1(\mathbf{y},\mathbf{u},\mathbf{v}) \end{pmatrix} = 0 \qquad (14)$$

Using the definition of $G_1$, we have that

$$G_1(G_{\mathbf{y}}^{-1}(\mathbf{x},\mathbf{u},\mathbf{v}),\mathbf{u},\mathbf{v}) = (x_2, x_3, \ldots, x_m, g_{m+1}(\mathbf{x},\mathbf{v}))$$

and therefore, $G_1(G_{\mathbf{y}}^{-1}(\mathbf{x},\mathbf{u},\mathbf{v}),\mathbf{u},\mathbf{v})$ does not depend on $\mathbf{u}$. Therefore,

$$D_{\mathbf{u}}G_1(G_{\mathbf{y}}^{-1}(\mathbf{x},\mathbf{u},\mathbf{v}),\mathbf{u},\mathbf{v}) = \begin{pmatrix} D_{\mathbf{y}}G_1(\mathbf{y},\mathbf{u},\mathbf{v})D_{\mathbf{u}}G_y^{-1}(\mathbf{x},\mathbf{u},\mathbf{v}) \\ +D_{\mathbf{u}}G_1(\mathbf{y},\mathbf{u},\mathbf{v}) \end{pmatrix} = 0 \qquad (15)$$

Now, if we take the partial derivative of (11) with respect to $\mathbf{u}$ we get

$$D_{\mathbf{y}}G(\mathbf{y},\mathbf{u},\mathbf{v})D_{\mathbf{u}}G_y^{-1}(\mathbf{x},\mathbf{u},\mathbf{v}) + D_{\mathbf{u}}G(\mathbf{y},\mathbf{u},\mathbf{v}) = 0$$

which can be written as

$$D_{\mathbf{u}}G_y^{-1}(\mathbf{x},\mathbf{u},\mathbf{v}) = -\left(D_{\mathbf{y}}G(\mathbf{y},\mathbf{u},\mathbf{v})\right)^{-1} D_{\mathbf{u}}G(\mathbf{y},\mathbf{u},\mathbf{v})$$

Since it was assumed that $-\left(D_{\mathbf{y}}G(\mathbf{y},\mathbf{u},\mathbf{v})\right)^{-1} D_{\mathbf{u}}G(\mathbf{y},\mathbf{u},\mathbf{v})$ is independent of the third variable, therefore $D_{\mathbf{u}}G_y^{-1}(\mathbf{x},\mathbf{u},\mathbf{v})$ is independent of $\mathbf{v}$ and $D_{\mathbf{u}}G_y^{-1}(\mathbf{x},\mathbf{u},\mathbf{v}) = D_{\mathbf{u}}G_y^{-1}(\mathbf{x},\mathbf{u},\mathbf{0})$. If that substitution is made in (15), we prove what we needed to show in (14). Finally, since $y[k] = g_1(\mathbf{y}[k],\mathbf{u}[k]) = x_1[k]$, then (13) is a state–space realization of the I/O map (6).

Second, we need to show that the realization in (13) is observable. To see that, notice that $h^m(\mathbf{x},\mathbf{v}) = G(\mathbf{y},\mathbf{u},\mathbf{v})$, with $\mathbf{x} = G(\mathbf{y},\mathbf{u},\mathbf{0})$. It now follows that $D_{\mathbf{x}}h^m(\mathbf{0},\mathbf{0})$ is equal to the identity matrix, which is full rank, thus proving the lemma. ■

Therefore, the necessary and sufficient conditions have been proven for an I/O map to have a state-space realization. However, since these conditions are difficult to

verify, [12] formulated a set of equivalent conditions that are easier to verify. These will be introduced next. Let $\gamma_j$ and $\gamma_{m+j}$ be defined by

$$\gamma_j(y_1,\ldots,y_m,u_1,\ldots,u_m) := D_{y_j}g(y_1,\ldots,y_m,u_1,\ldots,u_m)$$
$$\gamma_{m+j}(y_1,\ldots,y_m,u_1,\ldots,u_m) := D_{u_j}g(y_1,\ldots,y_m,u_1,\ldots,u_m)$$

and for $j=1,\ldots,m$ and define

$$\begin{aligned}\alpha_{i,j}(\mathbf{y},\mathbf{u},\mathbf{v}) &: = \gamma_j(y_i,\ldots,y_m,g_1(\mathbf{y},\mathbf{u},\mathbf{v}),\ldots,g_{i-1}(\mathbf{y},\mathbf{u},\mathbf{v}), \\ &\quad u_i,\ldots,u_m,v_1,\ldots,v_{i-1}) \qquad (16)\\ \beta_{i,j}(\mathbf{y},\mathbf{u},\mathbf{v}) &: = \gamma_{m+j}(y_i,\ldots,y_m,g_1(\mathbf{y},\mathbf{u},\mathbf{v}),\ldots,g_{i-1}(\mathbf{y},\mathbf{u},\mathbf{v}), \\ &\quad u_i,\ldots,u_m,v_1,\ldots,v_{i-1}) \qquad (17)\end{aligned}$$

Now, using the above definitions, the partial derivative of the $i$–th row of $G(\mathbf{y},\mathbf{u},\mathbf{v})$ with respect to $\mathbf{y}$ and $\mathbf{u}$ can be expressed as:

$$D_{\mathbf{y}}g_i = \sum_{j=1}^{i-1}\alpha_{i,m-i+j+1}D_{\mathbf{y}}g_j + \begin{bmatrix} 0 & \cdots & 0 & \alpha_{i,1} & \cdots & \alpha_{i,m-i+1}\end{bmatrix}$$
$$D_{\mathbf{u}}g_i = \sum_{j=1}^{i-1}\alpha_{i,m-i+j+1}D_{\mathbf{u}}g_j + \begin{bmatrix} 0 & \cdots & 0 & \beta_{i,1} & \cdots & \beta_{i,m-i+1}\end{bmatrix}$$

From the preceding equations, it can be seen that $D_{\mathbf{y}}G = U^{-1}L_\alpha$ and $D_{\mathbf{u}}G = U^{-1}L_\beta$, where

$$U = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ -\alpha_{2,m} & 1 & 0 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots \\ -\alpha_{m,2} & -\alpha_{m,3} & \cdots & -\alpha_{m,m} & 1 \end{bmatrix}$$

and

$$L_\alpha = \begin{bmatrix} \alpha_{1,1} & \alpha_{1,2} & \cdots & \alpha_{1,m} \\ 0 & \alpha_{2,1} & \cdots & \alpha_{2,m-1} \\ \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & \alpha_{m,1} \end{bmatrix},\ L_\beta = \begin{bmatrix} \beta_{1,1} & \beta_{1,2} & \cdots & \beta_{1,m} \\ 0 & \beta_{2,1} & \cdots & \beta_{2,m-1} \\ \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & \beta_{m,1} \end{bmatrix}$$

Thus the condition of lemma 5 for existence of a state-space realization is equivalent to $L_\alpha(\mathbf{y},\mathbf{u},\mathbf{v})$ being nonsingular and

$$M(\mathbf{y},\mathbf{u},\mathbf{v}) := [D_{\mathbf{y}}G(\mathbf{y},\mathbf{u},\mathbf{v})]^{-1}\, D_{\mathbf{u}}G(\mathbf{y},\mathbf{u},\mathbf{v}) = [L_\alpha(\mathbf{y},\mathbf{u},\mathbf{v})]^{-1}[L_\beta(\mathbf{y},\mathbf{u},\mathbf{v})]$$

independent of $\mathbf{v}$.

The existence of necessary and sufficient conditions implies that not every I/O map will be realizable. Therefore, a key point in [12] was to give a general form for the I/O equation that is guaranteed to satisfy the necessary and sufficient conditions for state-space realizability. This form is important, because a one of the new results of the thesis will be a more general form that also satisfies the realizability conditions. Equation (18) shows the form in [12], which was based on making every element of matrices $L_\alpha$ and $L_\beta$ independent of $\mathbf{v}$, therefore guaranteeing that matrix $M$ is independent of $\mathbf{v}$:

$$g(y_1,\ldots,y_m,u_1,\ldots,u_m) = \sum_{i=1}^{m-1} \breve{g}_i(y_i,y_{i+1},u_i) + \breve{g}_m(y_m,u_m) \tag{18}$$

where $\breve{g}_i$'s are smooth maps representing the partial maps of $g$.

The following algorithm was developed by Sadegh in [12] to check for state-space realizability of an I/O map:

**Algorithm 6**

**Data:** *The I/O map,* $g(y_1,\ldots,y_m,u_1,\ldots,u_m)$.

**Step 1:** *Compute scalars* $\alpha_{i,j}$ *and* $\beta_{i,j}$ *given by (16) and (17). Proceed to the next step only if* $\alpha_{i,1}(\mathbf{y},\mathbf{u},\mathbf{v}) \neq 0$ *on a neighborhood of the origin for* $i = 1,\ldots,m$. *Otherwise, the I/O map is not a DIOM.*

**Step 2:** *For* $i = 1,\ldots,m$ *and* $j = 2,\ldots,m-i+1$ *compute*

$$\bar{\alpha}_{i,j} := \frac{\alpha_{i,j}}{\alpha_{i,1}},\ \bar{\beta}_{i,j} = \frac{\beta_{i,j}}{\alpha_{i,1}}$$

**Step 3:** *For $k = 0, \ldots, m-1$ and $i = 1, \ldots, m-k$ compute*

$$\bar{M}_{i,i+k} = \bar{\beta}_{i,k+1} - \sum_{j=1}^{k} \bar{\alpha}_{i,j+1} \bar{M}_{i+j,i+k}$$

**Step 4:** *The I/O map is a DIOM if $D_{v_\ell} M_{i,j}(\mathbf{y}, \mathbf{u}, \mathbf{v}) = 0$, $\ell = 1, \ldots, i-1$, $\mathbf{v} = (v_1, \ldots, v_m)$.*

The next theorem was also given in [12] to summarize the main results. It will be given here to be used as a reference later.

**Theorem 7** *The I/O map $g(y_1, \ldots, y_m, u_1, \ldots, u_m)$ is a DIOM, i.e., has an observable state–space representation of order $m$, if and only if*

*i) $\alpha_{i,1}(\mathbf{y}, \mathbf{u}, \mathbf{v})$ is nonzero on a neighborhood of the origin for $i = 1, \ldots, m$.*

*ii) The functions $\bar{M}_{i,j}(\mathbf{y}, \mathbf{u}, \mathbf{v})$, $i = 2, \ldots, m$, $j = i, \ldots, m$, in algorithm 1 are independent of $v_1, \ldots, v_{i-1}$.*

The algorithm to be given next was also formulated in [12] to find the state-space realization from an I/O map once it has been determined that the map has a state-space realization (i.e. by using algorithm 1). The state-space realization given in (13) is still valid, but its computation can be difficult. Therefore the algorithm given next will give a step-by-step formulation of a state-space realization that is not hard to compute. To simplify the computation, first we need to examine the state equation given in Lemma (5)

$$\mathbf{x}[k+1] = f(\mathbf{x}[k], u[k]) = G_1(G_y^{-1}(\mathbf{x}[k], \mathbf{0}, \mathbf{0}), \mathbf{0}, \mathbf{v}[k])$$

Now, we define $\bar{\mathbf{y}} = G_y^{-1}(\mathbf{x}, \mathbf{0}, \mathbf{0})$, so the state vector $\mathbf{x} = G(\bar{\mathbf{y}}, 0, 0)$ and $f(\mathbf{x}, u) = G_1(\bar{\mathbf{y}}, 0, \mathbf{v})$. Now, using the definition of $G$ and $G_1$ given by (6) and (10), we are ready to present the algorithm.

**Algorithm 8** **Data:** *The I/O map, $g(y_1, \ldots, y_m, u_1, \ldots, u_m)$.*

**Step 1:** *Define the state variables,* $x_i[k]$, $i = 1, \ldots, m$, *in terms of the past inputs and outputs using the following recursive definition:*

$$x_i[k] = g(y[k-m+i-1], \ldots, y[k-1], x_1[k], \ldots, x_{i-1}[k], u[k-m+i-1], \ldots, u[k-1], 0, \ldots, 0)$$

**Step 2:** *Compute the* pseudo–outputs $\bar{y}_i$, $i = 1, \ldots, m$, *in terms of the state variables (numerically or symbolically) from the following equations:*

$$x_i = g(\bar{y}_i, \ldots, \bar{y}_m, x_1, \ldots, x_{i-1}, 0, \ldots, 0), i = 1, \ldots, m$$

**Step 3:** *Set the right hand side of state equations,* $x_i[k+1] = f_i(\mathbf{x}[k], u[k])$, $i = 1, \ldots, m$, *to*

$$f_i(\mathbf{x}, u) = g(\bar{y}_{i+1}, \ldots, \bar{y}_m, x_1, f_1(\mathbf{x}, u), \ldots, f_{i-1}(\mathbf{x}, u), 0, \ldots, 0, u, 0, \ldots, 0)$$

*where* $u$ *is the* $m - i + 1$*–th input element in* $g$.

## *2.6 An example*

In this section an example will be given to demonstrate the previously stated results. It will show how an I/O map can be checked to see if it satisfies the realizability conditions or not as well as how to find the state space realization from the I/O map.

Consider the I/O map given by

$$y[k] = g\left(y\left[k-3\right], y[k-2], y[k-1], u\left[k-3\right], [k-2], u[k-1]\right)$$

where

$$g(y_1, y_2, y_3, u_1, u_2, u_3) = (y_1+1)^{-3}(u_2+1)^3 + y_2 + (y_3+1)^3 - (u_1+1)^{-3}(u_2+1)^3 - (u_3+1)^{-3}$$

First, note that the equilibrium input and output are $\bar{u} = \bar{y} = 0$. Now, using equations

(16) and (17), the following can be calculated:

$$\begin{aligned}
\alpha_{1,1} &= -3(1+u_2)^3(1+y_1)^{-4} \\
\alpha_{1,2} &= 1 \\
\alpha_{1,3} &= 3(1+y_3)^2 \\
\alpha_{2,1} &= -3(1+u_3)^3(1+y_2)^{-4} \\
\alpha_{2,2} &= 1 \\
\alpha_{3,1} &= -3(1+v_1)^3(1+y_3)^{-4} \\
\beta_{1,1} &= 3(1+u_2)^3(1+u_1)^{-4} \\
\beta_{1,2} &= 3(1+u_2)^2\left[-(1+u_1)^{-3}+(1+y_1)^{-3}\right] \\
\beta_{1,3} &= 3(1+u_3)^{-4} \\
\beta_{2,1} &= 3(1+u_3)^3(1+u_2)^{-4} \\
\beta_{2,2} &= 3(1+u_3)^2\left[-(1+u_2)^{-3}+(1+y_2)^{-3}\right] \\
\beta_{3,1} &= 3(1+v_1)^3(1+u_3)^{-4}
\end{aligned}$$

Since $\alpha_{i,1} \neq 0$ in a neighborhood of the origin for $i = 1, \ldots, 3$, the first condition of Theorem 7 is satisfied. Using steps 2 and 3 of algorithm 6, the elements that need to be checked for $\mathbf{v}$ independence are:

$$\begin{aligned}
\bar{M}_{2,2} &= \frac{\beta_{2,1}}{\alpha_{2,1}} \\
\bar{M}_{3,3} &= \frac{\beta_{3,1}}{\alpha_{3,1}} \\
\bar{M}_{2,3} &= \frac{\beta_{2,2}}{\alpha_{2,1}} - \frac{\alpha_{2,2}}{\alpha_{2,1}}\frac{\beta_{3,1}}{\alpha_{3,1}}
\end{aligned}$$

It can be easily seen that $\bar{M}_{2,2}$, $\bar{M}_{3,3}$ and $\bar{M}_{2,3}$ are independent of $\mathbf{v}$. Therefore, $g(y_1, y_2, y_3, u_1, u_2, u_3)$ can be realized in state space form. Now, following Algorithm

6 we define the state variables to be

$$\begin{aligned} x_1[k] &= g(y[k-3], y[k-2], y[k-1], u[k-3], u[k-2], u[k-1]) \\ x_2[k] &= g(y[k-2], y[k-1], x_1[k], u[k-2], u[k-1], 0) \\ x_3[k] &= g(y[k-1], x_1[k], x_2[k], u[k-1], 0, 0) \end{aligned}$$

The psuedo-outputs in step 2 of Algorithm 8 are determined from

$$\begin{aligned} x_1 &= (\bar{y}_1 + 1)^{-3} + \bar{y}_2 + (\bar{y}_3 + 1)^3 \\ x_2 &= (\bar{y}_2 + 1)^{-3} + \bar{y}_3 + (x_1 + 1)^3 \\ x_3 &= (\bar{y}_3 + 1)^{-3} + x_1 + (x_2 + 1)^3 \end{aligned}$$

Solving the three equations recursively starting with the last one, we get

$$\begin{aligned} \bar{y}_3 &= \left(x_3 - x_1 - (x_2+1)^3\right)^3 - 1 \\ \bar{y}_2 &= \left(x_2 - \left(x_3 - x_1 - (x_2+1)^3\right)^3 + 1 - (x_1+1)^3\right)^3 - 1 \\ \bar{y}_1 &= \left( \begin{array}{c} x_1 - \left(x_2 - \left(x_3 - x_1 - (x_2+1)^3\right)^3 + 1 - (x_1+1)^3\right)^3 \\ - \left(x_3 - x_1 - (x_2+1)^3\right)^6 + 1 \end{array} \right)^3 - 1 \end{aligned}$$

Step 3 can finally be used to find the right hand side of the state equation

$$\begin{aligned} f_1 &= g(\bar{y}_2, \bar{y}_3, x_1, 0, 0, u) \\ f_2 &= g(\bar{y}_3, x_1, f_1, 0, u, 0) \\ f_3 &= g(x_1, f_1, f_2, u, 0, 0) \end{aligned}$$

Using this leads to:

$$\begin{aligned} f_1 &= (\bar{y}_2 + 1)^{-3} + \bar{y}_3 + (x_1+1)^3 - 1 - (u+1)^3 \\ f_2 &= (\bar{y}_3 + 1)^{-3} (u+1)^3 + x_1 + (f_1+1)^3 - (u+1)^3 - 1 \\ f_3 &= (x_1 + 1)^{-3} + f_1 + (f_2+1)^3 - (u+1)^{-3} - 1 \end{aligned}$$

Finally, after substitution for $\bar{y}_1$, $\bar{y}_2$, and $\bar{y}_3$ in terms of $x_1$, $x_2$, and $x_3$, the state equations become:

$$\begin{aligned}
x_1[k+1] &= x_2 - (u+1)^3 - 1 \\
x_2[k+1] &= \left(x_3 - x_1 - (x_2+1)^3\right)(u+1)^3 + x_1 + \left(x_2 - (u+1)^3\right)^3 - (u+1)^3 - 1 \\
x_3[k+1] &= (x_1+1)^{-3} + x_2 - (u+1)^3 + \left( \begin{array}{c} \left(x_3 - x_1 - (x_2+1)^3 - 1\right)(u+1)^3 \\ +x_1 + \left(x_2 - (u+1)^3\right)^3 \end{array} \right)^3 \\
&\quad - (u+1)^{-3} - 2
\end{aligned}$$

Thus, it was shown how the theory developed by Sadegh in [12] can be applied to find the state space realization from an I/O map when possible. The next chapter will present some simplifications of the necessary and sufficient conditions that will reduce the computation required, as well as present a new general class of I/O maps that are guaranteed to satisfy the realizability necessary and sufficient conditions. However, the next section will show first how to solve the realization problem for linear systems.

## *2.7 Linear State-Space Realization*

In this section, the state-space realization problem for linear systems will be examined. Following the procedure for the nonlinear case, we start with a linear state-space model and derive the input-output map. Once the form of the input-output map is found, we examine how to reverse the process and transform any input-output map of that specific form to a state-space model.

First consider a linear, SISO, observable, time-invariant state-space model given by

$$\mathbf{x}[k+1] = \mathbf{A}\mathbf{x}[k] + \mathbf{B}u[k] \tag{19}$$

$$y[k] = \mathbf{C}\mathbf{x}[k] \tag{20}$$

where $\mathbf{x}$ is the state vector, $\mathbf{A}$ is an $n \times n$ matrix, $\mathbf{B}$ is an $n \times 1$ vector, and $\mathbf{C}$ is a $1 \times n$ vector. A general formula for the solution of equation (19) can be derived easily as follows. If we start with initial condition $\mathbf{x} = \mathbf{x}[k]$ and apply (19) repeatedly, we get

$$\begin{aligned}
\mathbf{x}[k+1] &= \mathbf{A}\mathbf{x}[k] + \mathbf{B}u[k] \\
\mathbf{x}[k+2] &= \mathbf{A}\mathbf{x}[k+1] + \mathbf{B}u[k+1] \\
&= \mathbf{A}^2\mathbf{x}[k] + \mathbf{A}\mathbf{B}u[k] + \mathbf{B}u[k+1] \\
\mathbf{x}[k+3] &= \mathbf{A}\mathbf{x}[k+2] + \mathbf{B}u[k+2] \\
&= \mathbf{A}^3\mathbf{x}[k] + \mathbf{A}^2\mathbf{B}u[k] + \mathbf{A}\mathbf{B}u[k+1] + \mathbf{B}u[k+2], \\
&\vdots
\end{aligned}$$

In general, $\mathbf{x}[k+n]$ is given by

$$\mathbf{x}[k+n] = \mathbf{A}^n\mathbf{x}[k] + \mathbf{Q}\mathbf{u}[k] \tag{21}$$

where, $\mathbf{u}[k]$ is as defined in (3) and $\mathbf{Q}$ is given by

$$\mathbf{Q} = \begin{bmatrix} \mathbf{A}^{n-1}\mathbf{B} & \mathbf{A}^{n-2}\mathbf{B} & \cdots & \mathbf{A}\mathbf{B} & \mathbf{B} \end{bmatrix}$$

Similary, by applying (20) repeatedly, we get

$$\begin{aligned}
y[k] &= \mathbf{C}\mathbf{x}[k] \\
y[k+1] &= \mathbf{C}\mathbf{x}[k+1] \\
&= \mathbf{C}\mathbf{A}\mathbf{x}[k] + \mathbf{C}\mathbf{B}u[k] \\
y[k+2] &= \mathbf{C}\mathbf{x}[k+2] \\
&= \mathbf{C}\mathbf{A}^2\mathbf{x}[k] + \mathbf{C}\mathbf{A}\mathbf{B}u[k] + \mathbf{C}\mathbf{B}u[k+1] \\
y[k+3] &= \mathbf{C}\mathbf{x}[k+3] \\
&= \mathbf{C}\mathbf{A}^3\mathbf{x}[k] + \mathbf{C}\mathbf{A}^2\mathbf{B}u[k] + \mathbf{C}\mathbf{A}\mathbf{B}u[k+1] + \mathbf{C}\mathbf{B}u[k+2], \\
&\vdots
\end{aligned}$$

In general, we have the following relation

$$\mathbf{y}[k] = \mathbf{P}\mathbf{x}[k] + \mathbf{L}\mathbf{u}[k] \tag{22}$$

where, $\mathbf{y}[k]$ is as defined in (4), and $\mathbf{P}$ and $\mathbf{L}$ are given by

$$\mathbf{P} = \begin{bmatrix} \mathbf{C} \\ \mathbf{CA} \\ \vdots \\ \mathbf{CA}^{n-1} \end{bmatrix}$$

$$\mathbf{L} = \begin{bmatrix} 0 & \cdots & \cdots & \cdots & 0 \\ \mathbf{CB} & 0 & & & \vdots \\ \mathbf{CAB} & \mathbf{CB} & \ddots & & \vdots \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ \mathbf{CA}^{n-2}\mathbf{B} & \cdots & \cdots & \mathbf{CB} & 0 \end{bmatrix}$$

Notice that $\mathbf{P}$ is the observability matrix, which is full rank since the state-space model was assumed to be observable. Now, by solving for $\mathbf{x}[k]$ from (22), we get

$$\mathbf{x}[k] = \mathbf{P}^{-1}\left(\mathbf{y}[k] - \mathbf{L}\mathbf{u}[k]\right)$$

which, when substituted in (21), results in

$$\mathbf{x}[k+n] = \mathbf{A}^n\mathbf{P}^{-1}\left(\mathbf{y}[k] - \mathbf{L}\mathbf{u}[k]\right) + \mathbf{Q}\mathbf{u}[k]$$

or alternatively,

$$\begin{aligned} \mathbf{x}[k] &= \mathbf{A}^n\mathbf{P}^{-1}\left(\mathbf{y}[k-n] - \mathbf{L}\mathbf{u}[k-n]\right) + \mathbf{Q}\mathbf{u}[k-n] \\ &= \mathbf{A}^n\mathbf{P}^{-1}\mathbf{y}[k-n] + \left(\mathbf{Q} - \mathbf{A}^n\mathbf{P}^{-1}\mathbf{L}\right)\mathbf{u}[k-n] \end{aligned}$$

Finally, by substituting in (20) we get

$$y[k] = \mathbf{C}\mathbf{A}^n\mathbf{P}^{-1}\mathbf{y}[k-n] + \mathbf{C}\left(\mathbf{Q} - \mathbf{A}^n\mathbf{P}^{-1}\mathbf{L}\right)\mathbf{u}[k-n] \tag{23}$$

which is the input-output map for the state-space model in (19) and (20). Note that the structure of (23) implies that it can be written as

$$y[k] = \sum_{i=1}^{n} a_i y[k-i] + b_i u[k-i] \tag{24}$$

for some constants $a_i$ and $b_i$, $i = 1, \ldots, n$. We now derive a state-space realization corresponding to an input-output map of the form (24) using Algorithm 8. Define

$$g(y_1, \ldots, y_n, u_1, \ldots, u_n) = \sum_{i=1}^{n} a_{n-i+1} y_i + b_{n-i+1} u_i$$

then we have

$$y[k] = g(y[k-n], \ldots, y[k-1], u[k-n], \ldots, u[k-1])$$

From step 1 of the algorithm, we can define the state variables to be

$$\begin{aligned}
x_1[k] &= g(y[k-n], \ldots, y[k-1], u[k-n], \ldots, u[k-1]) \\
x_2[k] &= g(y[k-n+1], \ldots, y[k-1], x_1[k], u[k-n+1], \ldots, u[k-1], 0) \\
&\vdots \\
x_n[k] &= g(y[k-1], x_1[k], \ldots, x_{n-1}[k], u[k-1], 0, \ldots, 0)
\end{aligned}$$

From steps 2 and 3, we have

$$\begin{aligned}
x_2 &= g(\bar{y}_2, \ldots, \bar{y}_n, x_1, 0, \ldots, 0) \\
f_1(\mathbf{x}, u) &= g(\bar{y}_2, \ldots, \bar{y}_n, x_1, 0, \ldots, 0, u)
\end{aligned}$$

which implies that $f_1(\mathbf{x}, u) = x_2 + b_1 u$. Similarly,

$$\begin{aligned}
x_3 &= g(\bar{y}_3, \ldots, \bar{y}_n, x_1, x_2, 0, \ldots, 0) \\
f_2(\mathbf{x}, u) &= g(\bar{y}_3, \ldots, \bar{y}_n, x_1, f_1(\mathbf{x}, u), 0, \ldots, u, 0)
\end{aligned}$$

which implies that $f_2(\mathbf{x}, u) = x_3 + (a_1 b_1 + b_2) u$. Continuing this process, it can be seen that

$$\begin{aligned}
f_i(\mathbf{x}, u) &= x_{i+1} + \bar{b}_i u, \quad i = 2, \ldots, n-1 \\
f_n(\mathbf{x}, u) &= -\sum_{i=1}^{n} a_{n-i+1} x_i + \bar{b}_n u
\end{aligned}$$

where $\bar{b}_i = b_i + \sum_{j=1}^{i-1} a_{i-j}\bar{b}_j$. Finally, a state-space realization can be easily given by

$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ \vdots & & \ddots & \ddots & 0 \\ 0 & \cdots & \cdots & 0 & 1 \\ -a_1 & \cdots & \cdots & \cdots & -a_n \end{bmatrix}$$

$$\mathbf{B} = \begin{bmatrix} \bar{b}_1 \\ \vdots \\ \bar{b}_n \end{bmatrix}$$

# CHAPTER III

# NEW THEORETICAL RESULTS

This chapter will contain the main contributions of this thesis. It will present some new theoretical results that are mainly based on the theory presented in the last chapter. The chapter will be composed of two sections: The first section will present a simplification of the necessary and sufficient conditions for state-space realizability found by Sadegh in [12] (and re-introduced in the previous chapter); The second section will present a general class of I/O equations that are guaranteed to have a state-space realization. The class of I/O equations introduced in this chapter is more general than the one in (18) as will be shown later.

## *3.1 Necessary and Sufficient Conditions Simplification*

Recall from the last chapter that there exist two necessary and sufficient conditions for an I/O map to have a state-space realization:

*C1:* $L_\alpha(\mathbf{y}, \mathbf{u}, \mathbf{v})$ is invertible.

*C2:* $M(\mathbf{y}, \mathbf{u}, \mathbf{v})$ is independent of $\mathbf{v}$.

where,

$$L_\alpha(\mathbf{y},\mathbf{u},\mathbf{v}) = \begin{bmatrix} \alpha_{1,1} & \alpha_{1,2} & \cdots & \alpha_{1,m} \\ 0 & \alpha_{2,1} & \cdots & \alpha_{2,m-1} \\ \vdots & \ddots & \ddots & \ddots \\ 0 & 0 & \cdots & \alpha_{m,1} \end{bmatrix} \tag{25}$$

$$L_\beta(\mathbf{y},\mathbf{u},\mathbf{v}) = \begin{bmatrix} \beta_{1,1} & \beta_{1,2} & \cdots & \beta_{1,m} \\ 0 & \beta_{2,1} & \cdots & \beta_{2,m-1} \\ \vdots & \ddots & \ddots & \ddots \\ 0 & 0 & \cdots & \beta_{m,1} \end{bmatrix} \tag{26}$$

$$M(\mathbf{y},\mathbf{u},\mathbf{v}) = [L_\alpha(\mathbf{y},\mathbf{u},\mathbf{v})]^{-1}[L_\beta(\mathbf{y},\mathbf{u},\mathbf{v})] \tag{27}$$

In this section, it will be shown that *C1* and *C2* can be simplified. The simplification of *C2* results from the fact that certain elements $M(\mathbf{y},\mathbf{u},\mathbf{v})$ given by (27) are identical to some other elements of the same matrix, except that they are advanced a different number of times. Therefore, checking one element and the same element advanced differently for $\mathbf{v}$ independence is redundant and, as will be shown, it is enough to check the element that is advanced the most. A similar idea will be used to simplify *C1*. In order to simplify the proofs and the notation, arguments of functions will sometimes be omitted when they are obvious and can cause no confusion.

**Lemma 9** *Let*

$$M(\mathbf{y},\mathbf{u},\mathbf{v}) = \begin{bmatrix} M_{1,1} & \cdots & \cdots & M_{1,m} \\ 0 & \ddots & \ddots & M_{2,m-1} \\ \vdots & \ddots & \ddots & \ddots \\ 0 & \cdots & 0 & M_{m,1} \end{bmatrix}$$

*be as defined in (27) and let* $N(\mathbf{y},\mathbf{u},\mathbf{v}) = M(\mathbf{y},\mathbf{u},\mathbf{v})B$*, where* $B$ *is an* $m \times 1$ *matrix*

*given by*

$$B := \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix}$$

*. Then, $M(\mathbf{y}, \mathbf{u}, \mathbf{v})$ is independent of $\mathbf{v}$ if and only if $N(\mathbf{y}, \mathbf{u}, \mathbf{v})$ is independent of $\mathbf{v}$.*

Simply, the lemma says that matrix $M$ is independent of $\mathbf{v}$ if and only if its last column is independent of $\mathbf{v}$.

**Proof.** *Necessity:* It is trivial to see that if $M$ is independent of $\mathbf{v}$, then $N$ will also be independent of $\mathbf{v}$, because all elements of $N$ are elements of $M$. Specifically, $N = \begin{bmatrix} M_{1,m} & \cdots & M_{m,1} \end{bmatrix}^T$. Therefore, $N$ being independent of $\mathbf{v}$ is a necessary condition for $M$ to be independent of $\mathbf{v}$.

*Sufficiency:* First, *C2* is equivalent to the existence of a matrix $M(\mathbf{y}, \mathbf{u}, \mathbf{v})$, which is independent of $\mathbf{v}$, and satisfies

$$L_\alpha(\mathbf{y}, \mathbf{u}, \mathbf{v}) M(\mathbf{y}, \mathbf{u}, \mathbf{v}) = L_\beta(\mathbf{y}, \mathbf{u}, \mathbf{v}) \tag{28}$$

Notice that 28 is a system of $m(m+1)/2$ equations, which can be written as

$$\begin{bmatrix} \alpha_{1,1}M_{1,1} & \alpha_{1,1}M_{1,2} + \alpha_{2,1}M_{2,1} & \cdots & \sum_{i=0}^{m-1} \alpha_{1,1+i}M_{1+i,m-i} \\ 0 & \ddots & \ddots & \vdots \\ \vdots & & \alpha_{m-1,1}M_{m-1,1} & \alpha_{m-1,1}M_{m-1,2} + \alpha_{m-1,1}M_{m,1} \\ 0 & \cdots & 0 & \alpha_{m,1}M_{m,1} \end{bmatrix}$$

$$= \begin{bmatrix} \beta_{1,1} & \beta_{1,2} & \cdots & \beta_{1,m} \\ 0 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \beta_{m-1,1} & \beta_{m-1,2} \\ 0 & \cdots & 0 & \beta_{m,1} \end{bmatrix}$$

Due to the upper triangular nature of the above system of equations, the elements of the $M$ matrix can be found by solving for the elements of each column recursively

starting from the last nonzero element. For example, in the last column we have

$$\begin{aligned} M_{m,1} &= \frac{\beta_{m,1}}{\alpha_{m,1}} \\ M_{m-1,2} &= \frac{\beta_{m-1,2} - \frac{\alpha_{m-1,1}\beta_{m,1}}{\alpha_{m,1}}}{\alpha_{m-1,2}} \\ &\vdots \end{aligned}$$

A general formula for any element of $M$ is given by

$$M_{i,j} = \frac{\beta_{i,j} - \sum_{p=2}^{j} \alpha_{i,p} M_{i+p-1,j-p+1}}{\alpha_{i,1}} \tag{29}$$

for any $i = 1, \ldots, m$ and $j = 1, \ldots, m$. We will now set up an induction argument to prove that for any $i, j < m$, $M_{i+1,j} = \{M_{i,j}\}_1$. First, for $j = 1$ we have that

$$\begin{aligned} M_{i,1} &= \frac{\beta_{i,1}}{\alpha_{i,1}} \\ M_{i+1,1} &= \frac{\beta_{i+1,1}}{\alpha_{i+1,1}} \\ &= \left\{ \frac{\beta_{i,1}}{\alpha_{i,1}} \right\}_1 \\ &= \{M_{i,1}\}_1 \end{aligned}$$

where the equality $\frac{\beta_{i+1,1}}{\alpha_{i+1,1}} = \left\{ \frac{\beta_{i,1}}{\alpha_{i,1}} \right\}_1$ follows directly from the definitions in (16) and (17). Next, assume that for some $1 \leq j < m$ it is true that $M_{i+1,j-k} = \{M_{i,j-k}\}_1$ for $0 \leq k < j - 1$ and for any $1 \leq i < m$, we then show that for $j + 1$ the relation $M_{i+1,j+1} = \{M_{i,j+1}\}_1$ holds for any $1 \leq i < m$. Indeed, from (29) we see that

$$M_{i,j+1} = \frac{\beta_{i,j+1} - \sum_{p=2}^{j+1} \alpha_{i,p} M_{i+p-1,j-p+2}}{\alpha_{i,1}}$$

If we define $k = p - 2$, then we have

$$M_{i,j+1} = \frac{\beta_{i,j+1} - \sum_{p=2}^{j+1} \alpha_{i,p} M_{i+p-1,j-k}}{\alpha_{i,1}}$$

and

$$M_{i+1,j+1} = \frac{\beta_{i+1,j+1} - \sum_{p=2}^{j+1} \alpha_{i+1,p} M_{i+p,j-k}}{\alpha_{i+1,1}}$$

The assumption made in the induction argument allows us to write $M_{i+p,j-k}$ as $\{M_{i+p-1,j-k}\}_1$. Therefore, we get

$$\begin{aligned} M_{i+1,j+1} &= \frac{\beta_{i+1,j+1} - \sum_{p=2}^{j+1} \alpha_{i+1,p} M_{i+p,j-k}}{\alpha_{i+1,1}} \\ &= \frac{\{\beta_{i,j+1}\}_1 - \sum_{p=2}^{j+1} \{\alpha_{i,p}\}_1 \{M_{i+p-1,j-k}\}_1}{\{\alpha_{i,1}\}_1} \\ &= \left\{ \frac{\beta_{i,j+1} - \sum_{p=2}^{j+1} \alpha_{i,p} M_{i+p-1,j-k}}{\alpha_{i,1}} \right\}_1 \\ &= \{M_{i,j+1}\}_1 \end{aligned}$$

Next, we want to prove that if $M_{i+1,j}$ is independent of $\mathbf{v}$ then $M_{i,j}$ is also independent of $\mathbf{v}$ for $1 \leq i < m$ and $1 \leq j < m$. To do this, it will be easier to prove a logically equivalent statement: if $M_{i,j}$ is dependent on $\mathbf{v}$, then $M_{i+1,j}$ is also dependent on $\mathbf{v}$. This can be easily proven as follows. Assume that $M_{i,j}$ is dependent on $\mathbf{v}$ and let $k$ denote the highest subscript of $v$ in $M_{i,j}$. Then we have that $M_{i,j} = M_{i,j}(u_1, \ldots, u_m, y_1, \ldots, y_m, v_1, \ldots, v_k)$ and $\frac{\partial M_{i,j}}{\partial v_k} \neq 0$. As was proven earlier, $M_{i+1,j} = \{M_{i,j}\}_1$. So,

$$M_{i+1,j} = M_{i,j}(u_2, \ldots, u_m, v_1, y_2, \ldots, y_m, g(u_1, \ldots, u_m, y_1, \ldots, y_m), v_2, \ldots, v_{k+1})$$

and $\frac{\partial M_{i+1,j}}{\partial v_{k+1}} \neq 0$ implying that $M_{i+1,j}$ is dependent on $\mathbf{v}$.

Therefore, it is true that if $M_{i,j}$ is independent of $\mathbf{v}$ then $M_{i-k,j}$ is also independent of $\mathbf{v}$ for $1 < i \leq m$, $1 < j \leq m$, and $1 \leq k < i$. Since in the $M$ matrix, for any $1 \leq j \leq m$ the last column contains the element advanced the most, therefore, if the last column of $M$ is independent of $\mathbf{v}$ then $M$ is independent of $\mathbf{v}$.

Additionally, the first element of the last column of $N$, element $M_{1,m}$, need not be checked for $\mathbf{v}$ independence since the $\mathbf{v}$ independence of all the elements below it implies that it is independent of $\mathbf{v}$, which can be seen easily from (29). ■

Note that since $L_\alpha(\mathbf{y}, \mathbf{u}, \mathbf{v})$ is an upper triangular matrix, it will be invertible if and only if all its diagonal elements are nonzero. In [12], an algorithm (Algorithm 6) was given in which the *C1* was checked by checking every element along the diagonal of $L_\alpha(\mathbf{y}, \mathbf{u}, \mathbf{v})$. This can be slightly simplified using the following Lemma, which shows that if the first element along the diagonal of $L_\alpha(\mathbf{y}, \mathbf{u}, \mathbf{v})$ is nonzero, then the rest of the elements along the diagonal are nonzero.

**Lemma 10** *$L_\alpha(\mathbf{y}, \mathbf{u}, \mathbf{v})$ is invertible if and only if $\alpha_{1,1}(\mathbf{y}, \mathbf{u}, \mathbf{v}) = \frac{\partial g(y_1,\ldots,y_m,u_1,\ldots,u_m)}{\partial y_1} \neq 0$.*

**Proof.** *Necessity*: Since $L_\alpha(\mathbf{y}, \mathbf{u}, \mathbf{v})$ is upper triangular, if $\alpha_{1,1}(\mathbf{y}, \mathbf{u}, \mathbf{v}) = 0$ then the determinant of $L_\alpha(\mathbf{y}, \mathbf{u}, \mathbf{v}) = 0$ and therefore, $L_\alpha(\mathbf{y}, \mathbf{u}, \mathbf{v})$ is not invertible.

*Sufficiency*: An inductive argument similar to the one used in the last proof will be used here. First, note that $\alpha_{i,1} = \{\alpha_{i-1,1}\}_1$ for $i = 2, \ldots, m$. As the first step of the inductive argument, assume that $\alpha_{1,1} = \frac{\partial g(y_1,\ldots,y_m,u_1,\ldots,u_m)}{\partial y_1} \neq 0$. Now, for some $i = 1, \ldots, m-1$, let $\alpha_{i,1} \neq 0$; It will be shown that $\alpha_{i+1,1}$ is also nonzero. In general, $\alpha_{i,1} = \alpha_{i,1}(y_1, y_2 \ldots, y_{m-1}, y_m, \mathbf{u}, \mathbf{v})$. It's easy to see that

$$\begin{aligned} \alpha_{i+1,1} &= \{\alpha_{i,1}(y_1, y_2 \ldots, y_{m-1}, y_m, \mathbf{u}, \mathbf{v})\}_1 \\ &= \alpha_{i,1}(y_2, y_3 \ldots, y_m, g(y_1, \ldots, y_m, u_1, \ldots, u_m), \mathbf{u}, \mathbf{v}). \end{aligned}$$

In case $\alpha_{i,1}$ is independent of the $m$'th variable then $\alpha_{i,1} \neq 0$ directly implies that $\alpha_{i+1,1} \neq 0$, because advancing $\alpha_{i,1}$ in this case simply amounts to renaming the variables. Now consider the case where $\alpha_{i,1}$ is dependent on the $m$'th variable. Since it was assumed that $\frac{\partial g(y_1,\ldots,y_m,u_1,\ldots,u_m)}{\partial y_1} \neq 0$, then some $y_1$'s will be introduced in $\alpha_{i+1,1}$, which cannot be canceled (implying that $\alpha_{i+1,1} \neq 0$) since advancing $\alpha_{i,1}$ replaces any $y_1$ in $\alpha_{i,1}$ with a $y_2$. ■

Using Lemma 9 and Lemma 10, the following algorithm can be used to check whether an I/O map has a state space realization or not. This algorithm is simpler than the one given in [12], because it uses the above two Lemmas to simplify the realizability conditions.

**Algorithm 11**

**Data:** *The I/O map,* $g(y_1, \ldots, y_m, u_1, \ldots, u_m)$.

**Step 1:** *Compute scalars* $\alpha_{i,j}$ *and* $\beta_{i,j}$ *given by (16) and (17). Proceed to the next step only if* $\alpha_{1,1}(\mathbf{y}, \mathbf{u}, \mathbf{v}) \neq 0$ *on a neighborhood of the origin. Otherwise, the I/O map is not realizable in state space form.*

**Step 2:** *For* $i = 1, \ldots, m-1$, *compute the elements of the* $N$ *vector of Lemma 9 recursively starting from the last element and ignoring the first element (as was mentioned at the end of the proof of Lemma 9). The elements of* $N$ *can be calculated as follows:*

$$N(m-i) = \left[ \beta_{m-i+1,i} - \sum_{j=2}^{i} \alpha_{m-i+1,j} N(m-i+j-1) \right] / \alpha_{m-i+1,1}$$

**Step 3:** *The I/O map is a realizable in state space form if* $D_{\mathbf{v}} N(\mathbf{y}, \mathbf{u}, \mathbf{v}) = 0$, $\mathbf{v} = (v_1, \ldots, v_m)$.

The following Theorem summarizes the conditions for an I/O map to be realizable in state space form using the above two Lemmas to simplify Theorem 7.

**Theorem 12** *The I/O map* $g(y_1, \ldots, y_m, u_1, \ldots, u_m)$ *has an observable state–space representation of order* $m$, *if and only if*

*1)* $\alpha_{1,1}(\mathbf{y}, \mathbf{u}, \mathbf{v})$ *is nonzero on a neighborhood of the origin.*

*2) The vector* $N(\mathbf{y}, \mathbf{u}, \mathbf{v})$ *in algorithm 11 is independent of* $\mathbf{v}$.

## *3.2 General Class of Realizable I/O Maps*

In this section, a general class of I/O maps that is guaranteed to have a state space representation will be introduced. The I/O map class introduced here will also be more general than the one in 18.

**Theorem 13** *For any system of order m, any I/O map of the form*

$$g = f_m(u_m, y_m, f_{m-1}, \{f_{m-2}\}_1, \{f_{m-3}\}_2, \ldots, \{f_1\}_{m-2}) \tag{30}$$

*where,*

$$\begin{aligned} f_1 &= f_1(u_1, y_1, y_2) \\ f_k &= f_k\left(u_k, y_k, y_{k+1}, f_{k-1}, \{f_{k-2}\}_1, \ldots, \{f_1\}_{k-2}\right), \text{ for } k = 2, \ldots, m-1 \end{aligned}$$

*with the restriction that $\partial g/\partial y_1 \neq 0$ satisfies the realizability conditions 1) and 2) of Theorem 12.*

**Proof.** *1)* Since $\alpha_{1,1} = \partial g/\partial y_1$, then $\alpha_{1,1} \neq 0$ and the first condition of Theorem 12 is satisfied.

*2)* To show that the second condition is satisfied, an inductive argument based on generalizing the form given by (30) for a fixed, but arbitrary, system order will be constructed. To avoid confusion, increments in the induction argument will be denoted by superscripts. For example, the first step will be $g^1 = f_1(u_1, y_1, y_2)$, and the elements of the matrices $L_\alpha$, $L_\beta$, and $M$ will all have superscript 1.

As a first step in the inductive argument, let $g^1 = f_1(u_1, y_1, y_2)$ for some system of order $m \geq 2$. It is easy to see that in this case, $g^1$ is realizable since the matrices $L_\alpha$ and $L_\beta$, and therefore $M$, are independent of $\mathbf{v}$. Now, by generalizing $g$ to $g^2 = f_2(u_2, y_2, y_3, f_1(u_1, y_1, y_2))$, we have that the top $m-1$ rows of equation (28) are independent of $\mathbf{v}$, and so we need to check only $M^2_{m,1}$ for $\mathbf{v}$ independence. $M^2_{m,1}$ can

be easily calculated as follows:

$$\begin{aligned} M^2_{m,1} &= \frac{\left\{\frac{\partial g^2}{\partial u_1}\right\}_{m-1}}{\left\{\frac{\partial g^2}{\partial y_1}\right\}_{m-1}} \\ &= \frac{\left\{\frac{\partial f_2}{\partial f_1}\frac{\partial f_1}{\partial u_1}\right\}_{m-1}}{\left\{\frac{\partial f_2}{\partial f_1}\frac{\partial f_1}{\partial y_1}\right\}_{m-1}} \\ &= \frac{\left\{\frac{\partial f_1}{\partial u_1}\right\}_{m-1}}{\left\{\frac{\partial f_1}{\partial y_1}\right\}_{m-1}} \\ &= M^1_{m,1} \end{aligned}$$

Since $M^1_{m,1}$ is independent of $\mathbf{v}$, we have that $g^2$ is realizable and $M^2_{m,1} = M^1_{m,1}$. To set up an induction argument, assume that for some $2 \leq k \leq m-2$[1] it is true that

$$g^k = f_k\left(u_k, y_k, y_{k+1}, f_{k-1}, \{f_{k-2}\}_1, \ldots, \{f_1\}_{k-2}\right)$$

is realizable, and

$$\begin{aligned} M^k_{m,1} &= M^{k-1}_{m,1} = \ldots = M^1_{m,1} \\ M^k_{m-1,2} &= M^{k-1}_{m-1,2} = \ldots = M^2_{m-1,2} \\ &\vdots \\ M^k_{m-k+2,k-1} &= M^{k-1}_{m-k+2,k-1} \end{aligned}$$

It will be shown that $g^{k+1} = f_{k+1}\left(u_{k+1}, y_{k+1}, y_{k+2}, f_k, \{f_{k-1}\}_1, \ldots, \{f_1\}_{k-1}\right)$ is realizable and $M^{k+1}_{m-j+1,j} = M^k_{m-j+1,j}$ for $j = 1, \ldots, k$. Using equation (28) and the assumption that $g^k$ is realizable, there exist $M^k_{m,1}, \ldots, M^k_{m-k+1,k}$ independent of $\mathbf{v}$

[1] If $k$ is allowed to be $m-1$, then the proof is similar, but $g^{k+1} = f_{k+1}\left(u_{k+1}, y_{k+1}, f_k, \{f_{k-1}\}_1, \ldots, \{f_1\}_{k-1}\right)$.

that satisfy:

$$\begin{bmatrix} \left\{\frac{\partial f_k}{\partial y_1}\right\}_{m-k} & \cdots & \cdots & \left\{\frac{\partial f_k}{\partial y_k}\right\}_{m-k} \\ 0 & \left\{\frac{\partial f_k}{\partial y_1}\right\}_{m-k+1} & \cdots & \left\{\frac{\partial f_k}{\partial y_{k-1}}\right\}_{m-k+1} \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & \left\{\frac{\partial f_k}{\partial y_1}\right\}_{m-1} \end{bmatrix} \begin{bmatrix} M^k_{m-k+1,k} \\ \vdots \\ \vdots \\ M^k_{m,1} \end{bmatrix} = \begin{bmatrix} \left\{\frac{\partial f_k}{\partial u_k}\right\}_{m-k} \\ \vdots \\ \vdots \\ \left\{\frac{\partial f_k}{\partial u_1}\right\}_{m-1} \end{bmatrix} \tag{31}$$

By the induction hypothesis, equation (31) can be written as:

$$\begin{bmatrix} \left\{\frac{\partial f_k}{\partial y_1}\right\}_{m-k} & \cdots & \cdots & \left\{\frac{\partial f_k}{\partial y_k}\right\}_{m-k} \\ 0 & \left\{\frac{\partial f_{k-1}}{\partial y_1}\right\}_{m-k+1} & \cdots & \left\{\frac{\partial f_{k-1}}{\partial y_{k-1}}\right\}_{m-k+1} \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & \left\{\frac{\partial f_1}{\partial y_1}\right\}_{m-1} \end{bmatrix} \begin{bmatrix} M^k_{m-k+1,k} \\ \vdots \\ \vdots \\ M^k_{m,1} \end{bmatrix} = \begin{bmatrix} \left\{\frac{\partial f_k}{\partial u_k}\right\}_{m-k} \\ \vdots \\ \vdots \\ \left\{\frac{\partial f_1}{\partial u_1}\right\}_{m-1} \end{bmatrix} \tag{32}$$

Now, for $g^{k+1}$, equation (28) can be written as:

$$\begin{bmatrix} \alpha^{k+1}_{1,1} & \cdots & \alpha^{k+1}_{1,k+2} & 0 & \cdots & 0 \\ 0 & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & 0 \\ \vdots & & \ddots & \ddots & \ddots & \alpha^{k+1}_{m-k-1,k+2} \\ \vdots & & & \ddots & \ddots & \vdots \\ 0 & \cdots & \cdots & \cdots & 0 & \alpha^{k+1}_{m,1} \end{bmatrix} \begin{bmatrix} M^{k+1}_{1,m} \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ M^{k+1}_{m,1} \end{bmatrix} = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 0 \\ \beta^{k+1}_{m-k,k+1} \\ \vdots \\ \beta^{k+1}_{m,1} \end{bmatrix} \tag{33}$$

Notice that in equation (33), any rows advanced $m-k-1$ times or less will be independent of $\mathbf{v}$. If those rows are ignored, equation (33) becomes:

$$\begin{bmatrix} \left\{\frac{\partial f_{k+1}}{\partial y_1}\right\}_{m-k} & \cdots & \cdots & \left\{\frac{\partial f_{k+1}}{\partial y_k}\right\}_{m-k} \\ 0 & \left\{\frac{\partial f_{k+1}}{\partial y_1}\right\}_{m-k+1} & \cdots & \left\{\frac{\partial f_{k+1}}{\partial y_{k-1}}\right\}_{m-k+1} \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & \left\{\frac{\partial f_{k+1}}{\partial y_1}\right\}_{m-1} \end{bmatrix} \begin{bmatrix} M^{k+1}_{m-k+1,k} \\ \vdots \\ \vdots \\ M^{k+1}_{m,1} \end{bmatrix} = \begin{bmatrix} \left\{\frac{\partial f_{k+1}}{\partial u_k}\right\}_{m-k} \\ \vdots \\ \vdots \\ \left\{\frac{\partial f_{k+1}}{\partial u_1}\right\}_{m-1} \end{bmatrix} \tag{34}$$

so that $g^{k+1}$ is realizable if there exist $M^{k+1}_{m,1}, \ldots, M^{k+1}_{m-k+1,k}$ independent of $\mathbf{v}$ that satisfy equation (34). Now, The partials of $f_{k+1}$ with respect to $y$ on the left hand side of equation (34) can be written as:

$$\begin{aligned}
\frac{\partial f_{k+1}}{\partial y_1} &= \frac{\partial f_{k+1}}{\partial f_k}\frac{\partial f_k}{\partial y_1} \\
\frac{\partial f_{k+1}}{\partial y_2} &= \frac{\partial f_{k+1}}{\partial f_k}\frac{\partial f_k}{\partial y_2} + \frac{\partial f_{k+1}}{\partial \{f_{k-1}\}_1}\left\{\frac{\partial f_{k-1}}{\partial y_1}\right\}_1 \\
&\vdots \\
\frac{\partial f_{k+1}}{\partial y_k} &= \frac{\partial f_{k+1}}{\partial f_k}\frac{\partial f_k}{\partial y_k} + \frac{\partial f_{k+1}}{\partial \{f_{k-1}\}_1}\left\{\frac{\partial f_{k-1}}{\partial y_{k-1}}\right\}_1 + \ldots + \frac{\partial f_{k+1}}{\partial \{f_1\}_{k-1}}\left\{\frac{\partial f_1}{\partial y_1}\right\}_{k-1}
\end{aligned}$$

The partials of $f_{k+1}$ with respect to $\mathbf{u}$ can also be written in a similar manner. This shows that each row in equation (34) is a linear combination of the rows in equation (32) implying that $M^{k+1}_{m-j+1,j} = M^k_{m-j+1,j}$ for $j = 1, \ldots, k$ and that there exist $M^{k+1}_{m,1}, \ldots, M^{k+1}_{m-k+1,k}$ independent of $\mathbf{v}$ that satisfy equation (34). Therefore, $g^{k+1}$ is realizable and the form given by (30) satisfies the second condition of Theorem 12. ■

We now show how to construct a state-space model directly from the I/O map given in the theorem.

Using the I/O map given in 30, define the state variables as follows:

$$\begin{aligned}
x_1[k] &= f_m \\
x_2[k] &= \{f_{m-1}\}_1 \\
x_3[k] &= \{f_{m-2}\}_2 \\
&\vdots \\
x_m[k] &= \{f_1\}_{m-1}
\end{aligned}$$

Using the state variables defined above, a state-space model can be formulated:

$$\begin{aligned}
x_1[k+1] &= f_m(u, x_1, x_2, x_3, \ldots, x_m) \\
x_2[k+1] &= f_{m-1}(u, x_1, f_m(u, x_1, x_2, x_3, \ldots, x_m), x_3, x_4, \ldots, x_m) \\
x_3[k+1] &= f_{m-2}(u, x_1, f_m(u, x_1, x_2, x_3, \ldots, x_m), x_4, x_5, \ldots, x_m) \\
&\vdots \\
x_m[k+1] &= f_1(u, x_1, f_m(u, x_1, x_2, x_3, \ldots, x_m))
\end{aligned}$$

**Remark 14** *It should be noted that the I/O map class given by (30) is not the most general class of I/O maps that have a state-space realization. This can be easily shown by the following counter example. For a third order system, let the I/O map $g'$ be given by:*

$$g' = y_1 - (y_3 + u_2)u_1 - u_3 - y_2 \tag{35}$$

*It's easy to see that the $N$ vector of algorithm 11 is:*

$$N = \begin{bmatrix} -u_2 - y_2 + y_3 + u_2(y_1 - (y_3 + u_2)u_1 - u_3 - y_2) \\ -y_2 + y_3 + u_2(y_1 - (y_3 + u_2)u_1 - u_3 - y_2) \end{bmatrix}$$

*Therefore, the I/O map given by (35) is realizable in state-space form since $\frac{\partial g'}{\partial y_1} \neq 0$ and its $N$ vector is independent of $\mathbf{v}$. To see that $g'$ cannot be expressed in the form given by (30), first notice that for a third order system an I/O map of the form (30) is given by*

$$g = f_3(u_3, y_3, f_2(u_2, y_2, y_3, f_1(u_1, y_1, y_2)), f_1(u_2, y_2, y_3)) \tag{36}$$

*In this case, we have that*

$$\begin{aligned}
\frac{\frac{\partial g}{\partial u_1}}{\frac{\partial g}{\partial y_1}} &= \frac{\frac{\partial f_1(u_1, y_1, y_2)}{\partial u_1}}{\frac{\partial f_1(u_1, y_1, y_2)}{\partial y_1}} \\
&= F(u_1, y_1, y_2)
\end{aligned}$$

*for some function $F$. However, in case of the I/O map in (35), we have:*

$$\frac{\frac{\partial g}{\partial u_1}}{\frac{\partial g}{\partial y_1}} = -(y_3 + u_2)$$

*which cannot be expressed as $F(u_1, y_1, y_2)$ for some function $F$. Therefore, the realizable I/O map (35) cannot be represented as a map of the form in (30).*

As a final note of this chapter, it will be shown how the form (18) is a special case of (30). Indeed, if we define

$$\begin{aligned} f_1(x_1, x_2, x_3) &= \check{g}_1(x_2, x_3, x_1) \\ f_m(x_1, x_2, x_3, \ldots, x_{m+1}) &= \check{g}_m(x_2, x_1) + x_3 \\ f_k(x_1, x_2, x_3, \ldots, x_{k+4}) &= \check{g}_k(x_2, x_3, x_1) + x_4 \end{aligned}$$

for $1 < k < m$, then we get that the two forms are equivalent.

# CHAPTER IV

# AN APPLICATION

In this chapter, the previous theory will be applied to a hydraulic system to demonstrate how it can be applied in real life. First, the hydraulic system will be presented and then the state equations will be derived. Once the state-space model is found from first principles, it will be modeled in MATLAB and from that point on, it will be treated as a black box. In other words, the state-space model will be only used to simulate a real system so that input-output data can be obtained. Once the input-output data is obtained from MATLAB, an I/O map of the form (30) will be developed and its parameters will be estimated using the method of nonlinear least squares. When the I/O map is found, it will be used to find the state-space model for the system and then the output of the state-space model developed from the I/O map will be compared to the actual output from the Simulink model.

## *4.1 System Description and Modeling*

The hydraulic system considered in this chapter is a tank with a pressure accumulator and water flowing through the system as shown in Figure 1.

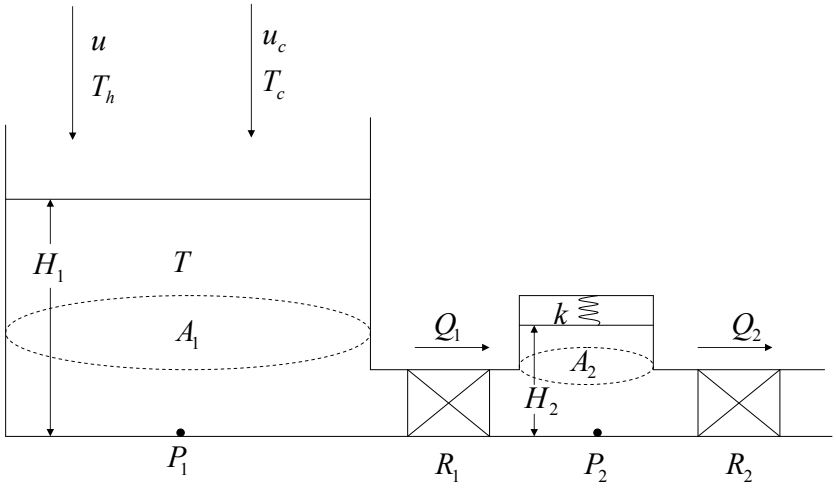


**Figure 1:** Hydraulic System

where,

$H_1$ = Height of the water in the tank

$H_2$ = Height of the water in the accumulator

$P_1$ = Pressure in the tank

$P_2$ = Pressure in the accumulator

$A_1$ = Cross sectional area of the tank

$A_2$ = Cross sectional area of the accumulator

$R_1$ = Fluid resistor between the tank and the accumulator

$R_2$ = Fluid resistor between the accumulator and the outside

$k$ = Spring constant of the spring in the accumulator

$u$ = Input volumetric flow rate of the hot water

$u_c$ = Input volumetric flow rate of the cold water

$Q_1$ = Volumetric flow rate across the first fluid resistor

$Q_2$ = Volumetric flow rate across the second fluid resistor

$T_h$ = Temperature of the input hot water

$T_c$ = Temperature of the input cold water

$T$ = Temperature of the water in the tank

### 4.1.1 System Operation and Assumptions

The system will be considered a single input-single output system. The input will be taken as the volumetric flow rate of the hot water going into the system and the output will be the temperature of the water in the tank. The temperature of the hot water as well as the flow rate and temperature of the cold water going into the system will be fixed. The initial temperature in the tank will be set so that it is between the cold and hot water temperatures. The first assumption that will be made is that the temperature is constant throughout the entire system (i.e. temperature in the tank is equal to the temperature in the accumulator). Second, it will be assumed that the temperature is not affected by the water in the accumulator, since the amount of water there will be considered negligible compared to the amount of water in the tank. Finally, the specific heat of water in the tank, which varies slightly as a function of temperature, will be assumed to be constant and equal to the specific heat of the hot and cold water inputs. These assumptions are not necessary, but they are mostly made to simplify the problem and emphasize the realization problem.

### 4.1.2 State-Space Model Derivation

In this section, the state-space model of the system will be derived so that it can be used to extract input-output data from MATLAB.

First, the differential equation describing the change of the height of the water in the tank can be written as

$$\frac{dH_1}{dt} = \frac{1}{A_1}\left(u + u_c - Q_1\right)$$

where, $\rho$ is the density of the water and $g$ is the acceleration due to gravity. Next, we have the following relation between the pressure and the height in the tank

$$P_1 = \rho g H_1$$

and the following relation between the flow rate across the first fluid resistor and the pressures in the tank and accumulator

$$Q_1 = \frac{P_1 - P_2}{R_1}$$

After substituting the above two relations in the differential equation for the height in the tank, we get

$$\frac{dP_1}{dt} = \frac{\rho g}{A_1}\left(u + u_c - \frac{P_1 - P_2}{R_1}\right) \tag{37}$$

Next, the differential equation describing the change of the height of the water in the accumulator is

$$\frac{dH_2}{dt} = \frac{1}{A_2}\left(Q_1 - Q_2\right)$$

and similarly, the following equation describes the relation between the height of the water in the accumulator and the pressure in the accumulator

$$P_2 = \left(\rho g + \frac{k}{A_2}\right) H_2$$

and the pressure in the accumulator and the flow rate across the second fluid resistor can be related by the following equation

$$Q_2 = \frac{P_2}{R_2}$$

Finally, after substituting the above two relations in the differential equation for the height of the water in the accumulator we get

$$\frac{dP_2}{dt} = \left(\frac{\rho g}{A_2} + \frac{k}{A_2^2}\right)\left(\frac{P_1 - P_2}{R_1} - \frac{P_2}{R_2}\right) \tag{38}$$

Note that it was assumed that the ambient pressure is zero since it acts on both the inlet and the outlet of the system, and therefore does not need to be included in the calculations as it cancels out.

Now, the differential equation governing how the temperature in the system changes will be derived. Since it was assumed that the temperature is constant throughout the system and is only affected by the water in the tank, to find the differential equation for the temperature, we only need to consider the energy conservation equations for the tank. Also, since the specific heat was assumed to be constant throughout the entire system, it will be denoted by $C_p$. The energy conservation equation of the tank is given by

$$\frac{d}{dt}\left(\rho A_1 H_1 C_p T\right) = \rho u C_p T_h + \rho u_c C_p T_c - Q_1 \rho C_p T$$

after canceling the constant terms from both sides and expressing the height in terms of pressure we get

$$\frac{A_1}{\rho g}\frac{d}{dt}\left(P_1 T\right) = u T_h + u_c T_c - Q_1 T$$

using the product rule results in the following equation

$$P_1 \frac{dT}{dt} + T\frac{dP_1}{dt} = \frac{\rho g}{A_1}\left(u T_h + u_c T_c - Q_1 T\right)$$

when $\frac{dP_1}{dt}$ is expressed using the expression from (37), the resulting equation becomes

$$P_1 \frac{dT}{dt} + T\frac{\rho g}{A_1}\left(u + u_c - \frac{P_1 - P_2}{R_1}\right) = \frac{\rho g}{A_1}\left(u T_h + u_c T_c - Q_1 T\right)$$

replacing $Q_1$ by $\frac{P_1 - P_2}{R_1}$ and rearranging results in

$$\frac{dT}{dt} = \frac{\rho g}{A_1 P_1}\left(u T_h + u_c T_c - T\left(u + u_c\right)\right) \tag{39}$$

We can now finally introduce the state-space model of the system. To do that we define the state variables $x_1, x_2$, and $x_3$ to be $T, P_1$, and $P_2$, respectively. Taking the output to be temperature and using equations (37), (38), and (39) results in the

following state model:

$$\begin{aligned}
\dot{x}_1 &= \frac{\rho g}{A_1 x_2}\left(uT_h + u_c T_c - x_1\left(u + u_c\right)\right) \\
\dot{x}_2 &= \frac{\rho g}{A_1}\left(u + u_c - \frac{x_2 - x_3}{R_1}\right) \\
\dot{x}_3 &= \left(\frac{\rho g}{A_2} + \frac{k}{A_2^2}\right)\left(\frac{x_2 - x_3}{R_1} - \frac{x_3}{R_2}\right) \\
y &= x_1
\end{aligned}$$

where, $y$ is the output. Since the work done in this thesis deals with discrete systems, the system will be discretized and used in the rest of the chapter. The discretized system can be represented as

$$\begin{aligned}
x_1[k+1] &= x_1[k] + T_s\frac{\rho g}{A_1 x_2[k]}\left(uT_h + u_c T_c - x_1[k]\left(u + u_c\right)\right) \\
x_2[k+1] &= x_2[k] + T_s\frac{\rho g}{A_1}\left(u + u_c - \frac{x_2[k] - x_3[k]}{R_1}\right) \\
x_3[k+1] &= x_3[k] + T_s\left(\frac{\rho g}{A_2} + \frac{k}{A_2^2}\right)\left(\frac{x_2[k] - x_3[k]}{R_1} - \frac{x_3[k]}{R_2}\right) \\
y[k] &= x_1[k]
\end{aligned} \tag{40}$$

where, $T_s$ is the sampling rate.

## *4.2 System Identification and Simulation*

In this section, the previously found state-space equations will be simulated in MATLAB to obtain input-output data. The system will then be treated as a black box and only the input-output data obtained will be used to recreate a state-space model for the system. This will be achieved in a few steps. First, an input-output map of the form (30) will be used to fit the data. Next, the state-space model corresponding to the input-output map will be formed. Finally, the results of the realized state-space model will be compared to the results from the actual system equations.

### 4.2.1 Data Generation

To generate the data, two MATLAB programs were written: play.m and simulation.m. Both programs can be found in the appendix. The purpose of the program play.m is to initialize the system parameters (i.e. spring constant, resistance values, etc...). In simulation.m, the state space model is ran for two hundred seconds with a sampling time of 0.1 seconds, which generates two vectors: an input vector and an output vector. Finally, the identification and simulation of the realized state-space model is done using another program, ssn.m. A more in depth discussion of the first two programs, which are used for data generation, will be given next

As was mentioned before, the purpose of the program play.m was to initialize the parameters used to get the input-output data. The following values were given to the parameters of the system (all units are SI units):

$\rho = 1000\ kg/m^3$

$g = 9.81\ m/s^2$

$H_{10} = 1\ m$

$H_{20} = 0.1\ m$

$A_1 = 3\ m^2$

$A_2 = 1\ m^2$

$R_1 = 100000\ m^4/Ns$

$R_2 = 150000\ m^4/Ns$

$k = 200\ Pa$

$T_h = 360\ K$

$Tc = 280\ K$

$T_0$=300 $K$

$u_c = 0.005\ m^3/s$

where, $H_{10}$ and $H_{20}$ are the initial heights in the tank and the accumulator, respectively, and $T_0$ is the initial temperature of the water in the tank. The initial

temperature was chosen to be 300 $K$, which is between the input hot and cold temperatures, which were chosen to be 360 $K$ and 280 $K$, respectively. It should be noted that since the purpose of this example is to illustrate how the state-space realization can be done for a real system, the above values were not chosen to specifically model any real system and that it was sufficient to choose realistic values for the parameters. Once the parameters are specified, the state space model (40) can be used to generate the input-output data.

To simulate the state-space model and generate the data, the program simulation.m was used. The program starts out by specifying the initial values for the states $x_1, x_2$, and $x_3$. It also initializes the time and the input and specifies the sampling rate. The input was chosen to be composed of the sum of four sine waves of different frequencies, 3, 4, 6, and 9. Each of the sine waves had a bias of 1 (i.e. mean value was 1) and the sum of the sine waves was multiplied by 0.005 to result in the mean value of the input to be 0.02 $m^3/s$. The reason for including four sine waves of different frequencies in the input was to make sure the input is rich enough to allow the identification to accurately identify the system. The bias was included to assure that the input flow rate is positive. Plots of the input used and the corresponding output are given next.

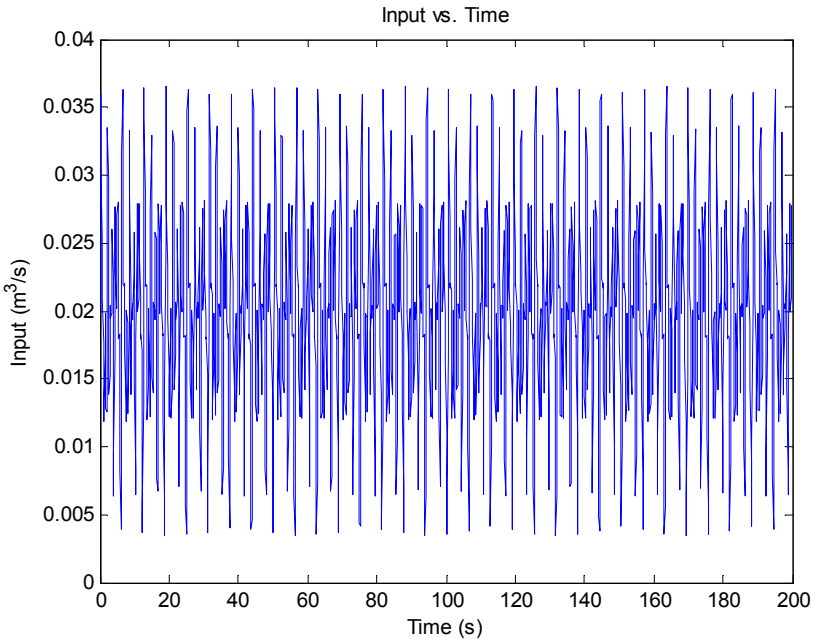


**Figure 2:** Input with four sine waves used for identification

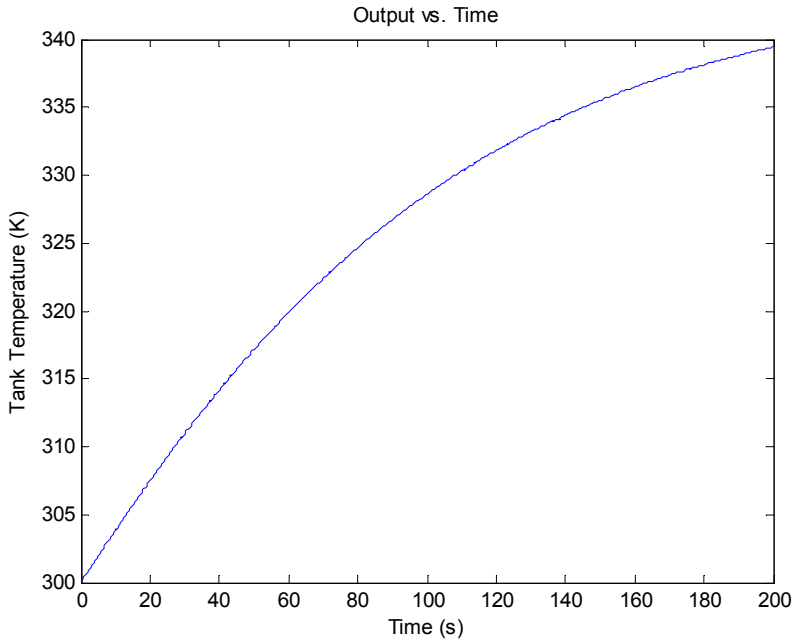


**Figure 3:** Output temperature of the tank used for identification

Even though they will not be used for the identification, the plots for the states $x_2$, and $x_3$, which correspond to the tank pressure and accumulator pressure, respectively will be shown to give a better picture of how the system behaves.

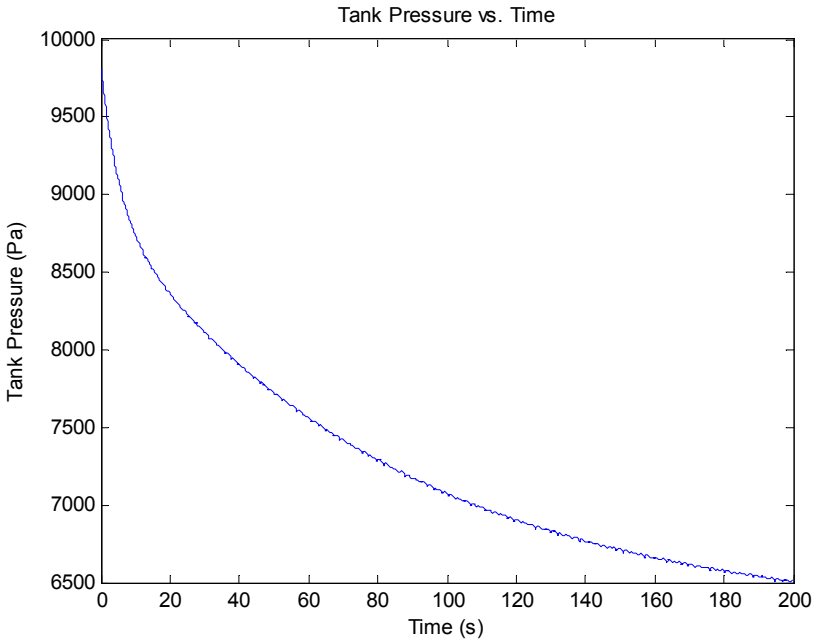


**Figure 4:** State variable $x_2$ (tank pressure)

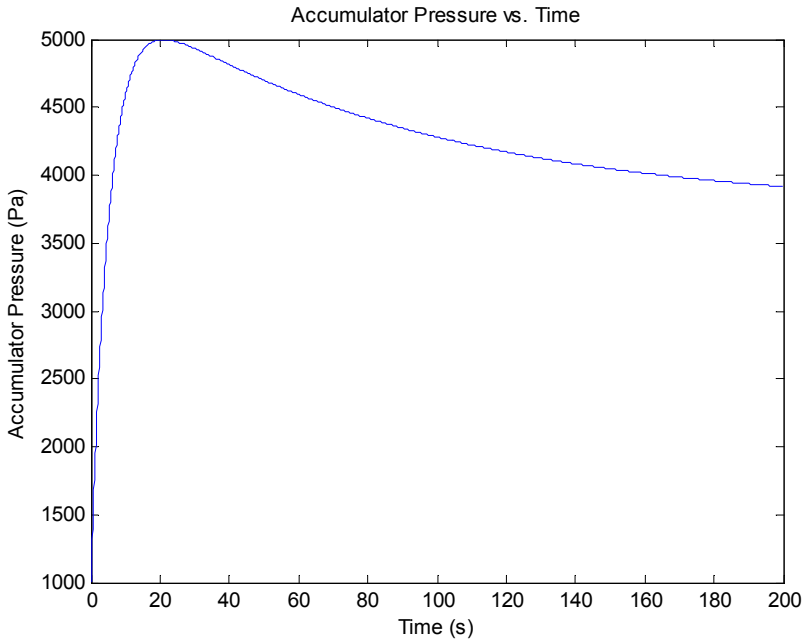


**Figure 5:** State variable $x_3$ (accumulator pressure)

### 4.2.2 Identification

To identify the system and apply the theory developed in earlier chapters, an input-output map of the form (30) must be found. For a third order system, (30) results in an equation of the form

$$y_4 = f_3(u_3, y_3, f_2(u_2, y_2, y_3, f_1(u_1, y_1, y_2)), f_1(u_2, y_2, y_3)) \tag{41}$$

Therefore, the functions $f_1, f_2$, and $f_3$ need to be identified. Once that is accomplished, a state-space model can be found easily using the technique given after the proof of Theorem 13, which shows that a state-space model corresponding to the input-output map (41) is given by

$$\begin{aligned} x_1[k+1] &= f_3(u, x_1, x_2, x_3) \\ x_2[k+1] &= f_2(u, x_1, f_3(u, x_1, x_2, x_3), x_3) \\ x_3[k+1] &= f_1(u, x_1, f_3(u, x_1, x_2, x_3)) \end{aligned} \tag{42}$$

In order to identify $f_1, f_2$, and $f_3$, some assumptions must be made about the form of these functions. First, $x_1$ will be taken as the temperature so that the output is $x_1$. It will also be assumed that $x_2$ is related to the pressure in the tank and $x_3$ is related to the pressure in the accumulator. With these assumptions, the following forms of $f_1, f_2$, and $f_3$ will be assumed:

$$\begin{aligned} f_1(x,y,z) &= c_1x + c_2y + c_3z \\ f_2(x,y,z,w) &= c_4x + c_5y + c_6z + c_7w \\ f_3(x,y,z,w) &= c_8x + c_9y + c_{10}/z + c_{11}xy + c_{12}x/z + c_{13}y/z \end{aligned} \tag{43}$$

where, $c_{1,\ldots}, c_{13}$ are constants to be determined. The reasoning for choosing the functions to be of the form (43) is as follows: First, since $x_2$ and $x_3$ are related to the pressures in the tank and accumulator, the dynamic equations for how they change in time will be linear, which is why $f_1$ and $f_2$ were chosen to be of a general linear form. Second, since $x_1$ is the temperature, we know that it changes nonlinearly in time as a function of the other state variables. Furthermore, we know that it depends on $1/x_2$ rather than on $x_2$, because increasing pressure in the tank implies increasing the height of the water which affects the temperature negatively). We also assume that the output temperature does not depend on the fluid in the accumulator. Therefore, the model chosen for $f_3$ is a bilinear model of the first two variables and the reciprocal of the third variable, but it is independent of the fourth variable.

Now that the form of the input-output map is chosen, we can proceed with the identification. The MATLAB program ssn.m was used to carry out the identification and simulation of the new state-space model that is developed from the input-output data. A discussion of how ssn.m works will be given next.

First, the program makes use of a user defined function called discretize.m (see appendix). The function discretize.m takes as inputs two column vectors (inputs and outputs) and outputs a matrix and another output vector. This is done to transform

the data into a form suitable for the identification. The rows of the matrix contain inputs and outputs at three consecutive time steps and the corresponding rows of the output vector are the outputs at the next time step. For example, if the inputs to the function discretize.m were two columns of inputs and outputs with $m$ steps, the output would be the matrix:

$$\begin{bmatrix} u_1 & u_2 & u_3 & y_1 & y_2 & y_3 \\ u_2 & u_3 & u_4 & y_2 & y_3 & y_4 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ u_{m-3} & u_{m-2} & u_{m-1} & y_{m-3} & y_{m-2} & y_{m-1} \end{bmatrix}$$

and the vector:

$$\begin{bmatrix} y_4 \\ y_5 \\ \vdots \\ y_m \end{bmatrix}$$

For the purposes of the identification, the matrix shown above is the "input" and the vector is the "output". The program then uses MATLAB function lsqcurvefit to identify the parameters of the system $(c_1, \ldots, c_{13})$ with an initial guess of 1 for all the parameters. Once the parameters are identified, they are used to simulate the new state space system given by (42) using the same input that was used for generating the input-output data.

For the above input and the parameters chosen earlier, the program gives the following coefficients (shown here only to four significant digits):

$c_1 = 0.4458$

$c_2 = 0.1478$

$c_3 = 0.08578$

$c_4 = 0.2042$

$c_5 = 0.07216$

$c_6$ = -0.005137

$c_7$ = -0.009454

$c_8$ = 11.46

$c_9$ = 0.9964

$c_{10}$ = -21.90

$c_{11}$ = -0.03022

$c_{12}$ = -2.197

$c_{13}$ = 0.1418

which result in the following approximation for the functions $f_1, f_2$, and $f_3$:

$$\begin{aligned} f_1(x,y,z) &= 0.4458x + 0.1478y + 0.08578z \\ f_2(x,y,z,w) &= 0.2042x + 0.07216y - 0.005137z - 0.009454w \\ f_3(x,y,z,w) &= 11.46x + 0.9964y - 21.90/z - 0.03022xy - 2.197x/z + 0.1418y/z \end{aligned}$$

Finally, this leads to the following state-space model:

$$\begin{aligned} x_1[k+1] &= 11.46u + 0.9964x_1 - 21.90/x_2 - 0.03022ux_1 - 2.197u/x_2 \\ &\quad +0.1418x_1/x_2 \\ x_2[k+1] &= 0.1453u + 0.06704x_1 + 0.1125/x_2 + 0.0001552ux_1 + .01129u/x_2 \\ &\quad -0.0007284x_1/x_2 - 0.009454x_3 \\ x_3[k+1] &= 1.429u + 0.2333x_1 - 1.879/x_2 - 0.002592ux_1 - 0.1885u/x_2 \\ &\quad +0.1216x_1/x_2 \end{aligned} \tag{44}$$

After the program approximates the coefficients, it uses them with the functions $f_1, f_2$, and $f_3$ to calculate the states recursively. The output of the actual system as well as the approximate state-space model are shown on the next plot

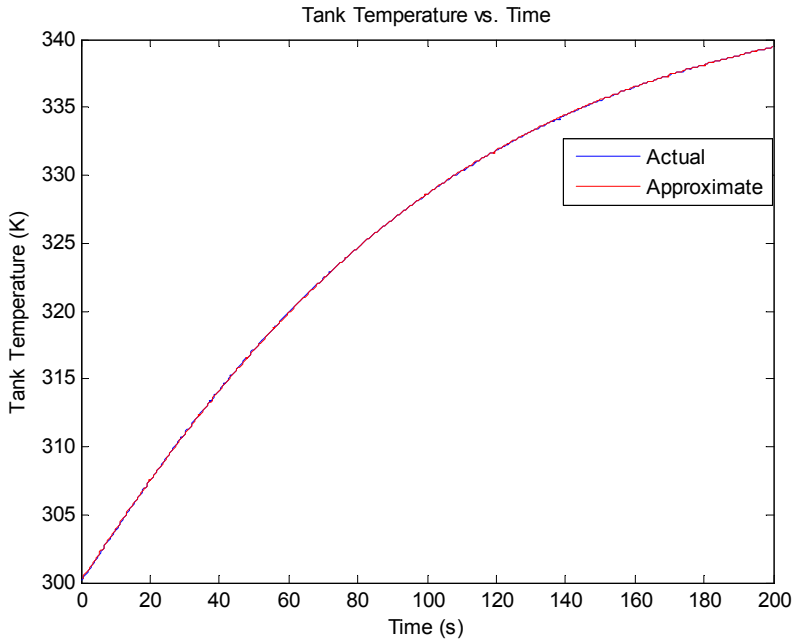


**Figure 6:** Comparison of outputs between the actual system and the realized state-space model

As can easily be seen, the output of the actual system and the realized state-space model match very well. The absolute mean error between the temperatures is only 0.02589 $K$, approximately. The next figure shows the error between the two outputs.

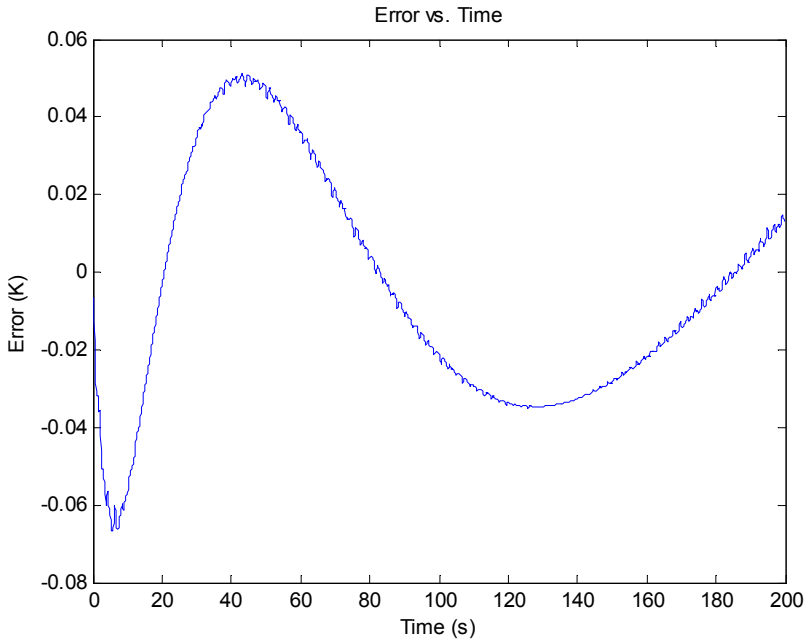


**Figure 7:** Difference between actual output and output from realized state-space model

Therefore, it can be seen that the output of the state-space realization developed approximates the output of the system very closely. However, the above plots were shown only for the time period that was used in the identification. In order to examine how the system behaves when it is outside of the data used for the identification, the actual system was ran for 400 seconds and the realized system was also ran for 400 seconds using the previously estimated coefficients and without going through the identification again. Two plots for the comparison of the actual and realized system output follow.

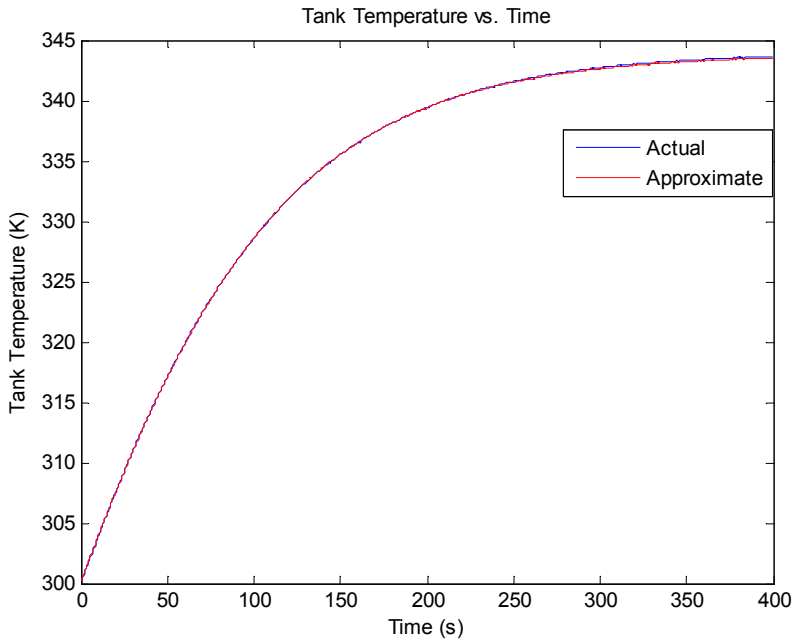


**Figure 8:** Output temperatures of the actual and realized system when both are ran for 400 seconds

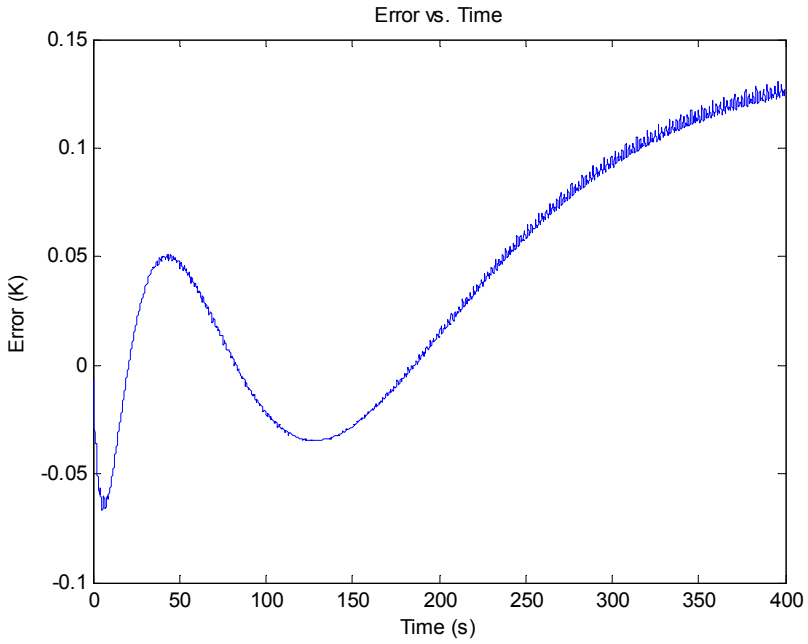


**Figure 9:** Difference between outputs of the actual and the realized model when both are ran for 400 seconds

Obviously, the error increases, but still not by much. The absolute mean error was approximately 0.05559 $K$, which is very good considering that it was generated from a data set that is twice as long as the one used for the identification.

## *4.3 Theoretical Check*

In this section it will be analytically checked whether the input-output map of the system introduced in the first section can be represented in the form given by (30) or not. It should be stressed that the work in this section is not needed to carry out the objectives of the chapter. Even if the system cannot be represented in the form (30), an input-output map of the form (30) can still be used to identify the system, because any identification procedure will inherently still be an approximation of the actual system. However, the results in this section will serve to show the generality of

the form in (30) and will also help to show what criteria causes systems to not be of the form in (30). Due to the complexity that results from carrying out the following procedure, the details will not be explicitly shown. Instead, a step-by-step explanation of the procedure will be given and the results from a Mathematica Notebook, which can be found in the appendix, will be shown. The ultimate objective will be to derive the functions $f_1, f_2$, and $f_3$ of (41) analytically. In the following derivations, variables will only be advanced once or de-advanced once. Therefore, to simplify the notation, if a variable $x$ is advanced once, it will be written as $x^+$ and if it is de-advanced once it will be written as $x^-$. Alternatively, if the current time step is $k$, a superscript $+$ is another way of saying the variable is evaluated at the time step $k+1$ and a superscript $-$ implies the variable is evaluated at the time step $k-1$. If a variable has no superscript then it is evaluated at the current time step, $k$.

The main idea of this section will be to examine when a general third order state-space model can be put in the form

$$x_1^+ = f_3(u, x_1, x_2, x_3) \tag{45}$$

$$x_2^+ = f_2(u, x_1, f_3(u, x_1, x_2, x_3), x_3) \tag{46}$$

$$x_3^+ = f_1(u, x_1, f_3(u, x_1, x_2, x_3)) \tag{47}$$

and to carry out the transformation from the general model to the one given above. Once that is done, the functions $f_1, f_2$, and $f_3$ will be readily available.

A general third order state-space model can be represented as

$$x_1^+ = h_1(u, x_1, x_2, x_3) \tag{48}$$

$$x_2^+ = h_2(u, x_1, x_2, x_3) \tag{49}$$

$$x_3^+ = h_3(u, x_1, x_2, x_3) \tag{50}$$

which can be written as

$$\mathbf{x}^+ = H(u, \mathbf{x})$$

where $\mathbf{x} = \begin{bmatrix} x_1 & x_2 & x_3 \end{bmatrix}^T$. Assuming that the jacobian of $H$ with respect to $\mathbf{x}$ is nonzero, we can solve for $\mathbf{x}$:

$$\mathbf{x} = H^{-1}(u, \mathbf{x}^+)$$

By de-advancing the previous equation, we get

$$\mathbf{x}^- = H^{-1}(u^-, \mathbf{x})$$

which implies that

$$x_1^- = h_1^{-1}(u^-, x_1, x_2, x_3)$$

Finally, if it is possible, for some functions $\bar{h}_1$ and $\bar{h}_4$, to write the previous equation as

$$x_1^- = \bar{h}_1(u^-, x_1, \bar{h}_4(x_2, x_3)),$$

where $\frac{\partial \bar{h}_1}{\partial \bar{h}_4} \neq 0$, then we can obtain the function $f_1$ of form (30). To see this, define a new state variable

$$\hat{x}_3 = \bar{h}_4(x_2, x_3) \tag{51}$$

Then we can write $x_1^-$ as

$$x_1^- = \bar{h}_1(u^-, x_1, \hat{x}_3)$$

and after advancing once and solving for $\hat{x}_3^+$, we get

$$\hat{x}_3^+ = f_1(u, x_1, x_1^+) \tag{52}$$

To get $f_3$, solve for $x_3$ from equation (51):

$$x_3 = k(x_2, \hat{x}_3)$$

and substitute in equations (48) and (49) to get

$$x_1^+ = h_1(u, x_1, x_2, k(x_2, \hat{x}_3)) \tag{53}$$

$$x_2^+ = h_2(u, x_1, x_2, k(x_2, \hat{x}_3)) \tag{54}$$

Now, simply rewrite equation (53) as

$$x_1^+ = f_3(u, x_1, x_2, \hat{x}_3) \tag{55}$$

to obtain the function $f_3$. To obtain $f_2$, the previous equation must be solved for $x_2$. If that is possible, then $x_2$ can be re-written as

$$x_2 = f_3^{-1}(u, x_1, x_1^+, \hat{x}_3)$$

which must then be substituted in equation (54) to get

$$x_2^+ = h_2(u, x_1, f_3^{-1}(u, x_1, x_1^+, \hat{x}_3), k(x_2, \hat{x}_3))$$

Finally, rewrite the last equation as

$$x_2^+ = f_2(u, x_1, x_1^+, \hat{x}_3)$$

to obtain $f_2$.

Hence, if all the previous steps are possible, then a third order system will have an input-output equation of the form (30). As was seen in a previous chapter and by the conditions in the above derivation it will not, in general, be possible to do this for any system. However, the conditions are not very strict and mostly relate to functions being invertible. In fact, it was possible to derive the functions $f_1, f_2$, and $f_3$ analytically for the hydraulic system presented in the first section of this chapter using the previous derivation. The functions are given next.

Using the above derivation, it was found that $f_1, f_2$, and $f_3$ can be given by:

$$
\begin{aligned}
f_1(x,y,z) &= -\frac{c_1c_2R_1x}{c_1(c_2+c_3)-c_2R_1} + \frac{c_1(-T_cu_c - T_hx + y(u_c+x))}{y-z} \\
f_2(x,y,z,w) &= \frac{\begin{pmatrix} c_1^2(T_cu_c + T_hx - (u_c+x)y) + c_2R_1w(-y+z) \\ -c_1(-(c_2+c_3)w(y-z) \\ +R_1((1+c_2)T_cu_c + T_hx + (u_c+x)(-2y+z) \\ +c_2(T_hx - xy - u_cz))) \end{pmatrix}}{R1(y-z)} \\
f_3(x,y,z,w) &= y + \frac{c_1(T_cu_c + T_hx - (u_c+x)y)}{z}
\end{aligned}
$$

where,

$$
\begin{aligned}
c_1 &= \frac{T_s\rho g}{A_1} \\
c_2 &= 1 - T_s\left(\frac{\rho g}{A_2} + \frac{k}{A_2^2}\right)\left(\frac{1}{R_1} + \frac{1}{R_2}\right) \\
c_3 &= \frac{T_s}{R_1}\left(\frac{\rho g}{A_2} + \frac{k}{A_2^2}\right)
\end{aligned}
$$

If $f_1, f_2$, and $f_3$ are as defined above, the following plot gives the error between the output of the realized state-space model (equations 45, 46, and 47) and the actual output of the system.

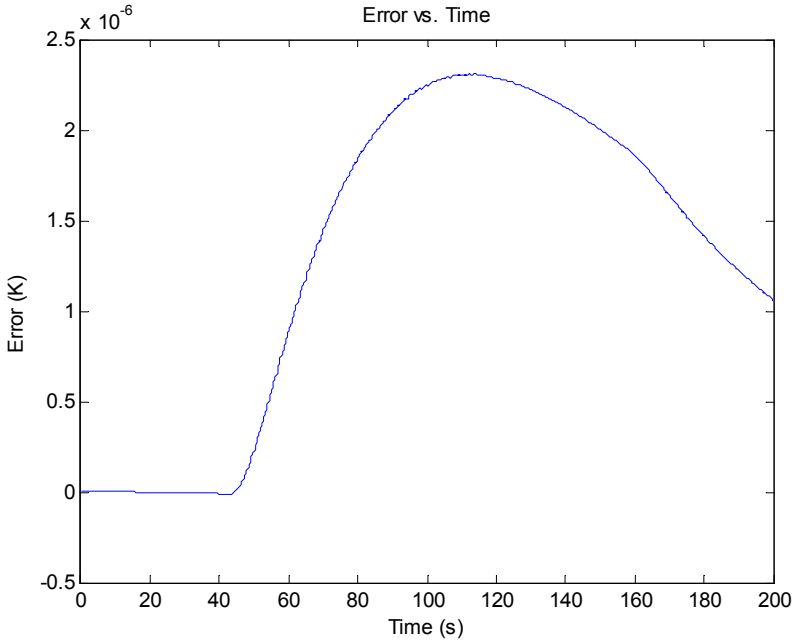


**Figure 10:** Output Error of a state-space realization

Notice that the scale on the y-axis is multiplied by $10^{-6}$. The absolute mean error was only $1.3202 \times 10^{-6}K$, which is small enough to be attributed to round-off errors. Another thing to note is that the functions $f_1, f_2$, and $f_3$, are not unique. An easy way to see that is from equation (51), because if a different function is chosen for $\hat{x}_3$, then it will ultimately lead to different forms of $f_1, f_2$, and $f_3$. However, some forms have less round-off errors than others.

# CHAPTER V

# CONCLUSIONS AND FURTHER WORK

In the previous chapters, the topic of state-space realization of input-output maps was examined in detail. First, some background information about the realization problem was given in chapter 2 and then chapter 3 introduced some new theoretical results, which were then applied to model a hydraulic system in chapter 4. The main contributions of this thesis are the results presented in chapter 3.

In chapter 3, two main results were proved. The first shows that the necessary and sufficient conditions for realizability of input-output maps of nonlinear systems can be simplified. The importance of this result is mainly due to the reduction in computation needed to check whether a given input-output map is realizable or not. It can also be used to simplify proofs, especially when proving that certain forms of input-output equations are always realizable, as was done in the proof of the second result. The second result in chapter 3 is the most important result in the thesis. It introduces a very general form, one that is much more general than existing ones in the literature, for input-output maps and proves that it is always realizable. The importance of this result is twofold. First, the result can be applied directly when identifying systems, because instead of using generic models (such as polynomial or bilinear models) which may not be realizable, a model of the form introduced can be used, which leads to the state-space of the identified system to be readily available from the input-output map directly as was shown in chapter 4. Second, the class of input-output maps given in chapter 3 is a big step towards finding the most general input-output map form that satisfies the realizability conditions. It was shown in chapter 3, by means of a counter example, that the form introduced is not the most

general one. However, finding the most general form would be a very powerful and fundamental step in control theory. This is the main area of research in which further work can be done.

The most fundamental fact that makes finding the most general form very powerful is that any physical system has a state-space representation. Although in some cases it may be very hard, or impossible, to find the exact state-space representation of a given system, each system has state-variables and their behavior can be described by first order differential equations in time. Therefore, every physical system has an input-output map that is realizable. Due to this fact, if the most general form is found then given any black box system, we can find a state-space representation for it. Of course, this assumes that we are able to identify the system using a model of the form found. This would lead to a new subject in system identification research: how to identify systems using input-output models of the most general form that satisfy the realizability conditions. It would not make much sense to use models that are not realizable since we know (even before identifying the parameters) that it is impossible for the actual state-space model to have an input-output equation that is not realizable.

Hence, the main result presented in this paper is a big step towards achieving the ultimate goal in control theory: given a black box system, identify it and control it. The two fields of system identification and system control evolved into two very large and mature fields, mostly independently. Finding the most general form of input-output maps that is realizable would link the two fields, since system identification is mostly concerned with finding input-output maps and much of control theory is based on state-space representations of systems.

# APPENDIX A

# MATLAB AND MATHEMATICA PROGRAMS

*play.m:*

```
clear all;
clc;
ro = 1000;
g = 9.81;
h10 = 1;
h20 = 0.1;
A1 = 3;
A2 = 1;
R_1 = 100000;
R2 = 150000;
k = 200;
Th = 360;
Tc = 280;
Uc = 0.005;
T0=300;
```

*simulation.m*

```
x2(1) = ro*g*h10;
x3(1) = ro*g*h20 + (k*h20)/A2;
x1(1) = T0;
Time(1) = 0;
u(1) = 0.005*(4+sin(3*Time(1))+sin(6*Time(1))+sin(9*Time(1))...
```

```
+sin(4*Time(1)));
Ts = 0.1;
for i = 1:200/Ts-1
Time(i+1) = Time(i)+Ts;
u(i+1) = 0.005*(4+sin(3*Time(i+1))+sin(6*Time(i+1))+sin(9*Time(i+1))...
+sin(4*Time(i+1)));
x1(i+1) = x1(i) + (Ts*ro*g/(A1*x2(i)))*(u(i)*Th+Uc*Tc-x1(i)*(u(i)+Uc));
x2(i+1) = x2(i) + (Ts*ro*g/A1)*(u(i)+Uc-(x2(i)-x3(i))/R_1);
x3(i+1) = x3(i) + Ts*(ro*g/A2+k/(A2^2))*((x2(i)-x3(i))/R_1-x3(i)/R2);
end
```

*ssn.m:*

```
in = u';out = x1';
[Xdata,Ydata] = discretize(in,out);
C(1:13)=1;
options=optimset('lsqcurvefit');
options.MaxFunEvals=100000;
options.MaxIter=10000;
options.TolFun=1e-8;
options.TolX=1e-8;
[c,RES] = lsqcurvefit(@Non_recursive_func,C,Xdata,Ydata,[],[],options);
x1n(1) = fthree(u(3),x1(3),ftwo(u(2),x1(2),x1(3),fone(u(1),x1(1),x1(2),c),c)...
          ,fone(u(2),x1(2),x1(3),c),c);
x2n(1) = ftwo(u(3),x1(3),x1n(1),fone(u(2),x1(2),x1(3),c),c);
x3n(1) = fone(u(3),x1(3),x1n(1),c);
Time(1) = 0;
for i = 1:200/Ts-4
Time(i+1) = Time(i)+Ts;
```

```
x1n(i+1) = fthree(u(i+3),x1n(i),x2n(i),x3n(i),c);
x2n(i+1) = ftwo(u(i+3),x1n(i),x1n(i+1),x3n(i),c);
x3n(i+1) = fone(u(i+3),x1n(i),x1n(i+1),c);
end
plot(x1(4:end)-x1n)
figure
plot(x1)
hold on
plot(x1n,'r')
hold off
mean(abs((x1(4:end)-x1n)))
```

*discretize.m:*

```
function [IN,OUT] = discretize(u,y)
a_vec = size(u);
a = a_vec(1,1);
if mod(a,3) == 1
offset = 4;
offsety = 1;
else
offset = 3;
offsety = 0;
end
X(:,1) = u(1:end-offset);
X(:,2) = u(2:end-offset+1);
X(:,3) = u(3:end-offset+2);
X(:,4) = y(1:end-offset);
X(:,5) = y(2:end-offset+1);
```

```
X(:,6) = y(3:end-offset+2);

IN = X;

OUT = y(4:end-offsety);
```

*Non_recursive_func.m:*

```
function out = Non_recursive_func(c,X)

u1 = X(:,1);

u2 = X(:,2);

u3 = X(:,3);

y1 = X(:,4);

y2 = X(:,5);

y3 = X(:,6);

out = fthree(u3,y3,ftwo(u2,y2,y3,fone(u1,y1,y2,c),c),fone(u2,y2,y3,c),c);
```

*fone.m:*

```
function a = fone(x,y,z,c)

a = c(1)*x + c(2)*y + c(3)*z;
```

*ftwo.m:*

```
function a = ftwo(x,y,z,w,c)

a = c(4)*x + c(5)*y + c(6)*z + c(7)*w;
```

*fthree.m:*

```
function a = fthree(x,y,z,w,c)

a = c(8)*x + c(9)*y + c(10)./z + c(11)*x.*y + c(12)*x./z + c(13)*y./z;
```

*Mathematica Notebook used to generate $f_1, f_2,$ and $f_3$ :*

```
Clear["Global`*"]
(*
x1p = x1+(Ts*ro*g/(A1*x2))*(U*Th+Uc*Tc-x1(U+Uc));
x2p = x2+ Ts*(ro*g/A1)*(U+Uc-(x2-x3)/R1);
x3p = x3 + Ts*(ro*g/A2+k/(A2^2))*((x2-x3)/R1-x3/R2);
*)
dAd = {x1 → x1m, x2 → x2m, x3 → x3m, U → Um, x1p → x1, x2p → x2, x3p → x3, Up → U};
Ad = {x1m → x1, x2m → x2, x3m → x3, Um → U, x1 → x1p, x2 → x2p, x3 → x3p, U → Up, x3h → x3hp};
temp1 = x1 + (c1 / x2) * (U * Th + Uc * Tc - x1 (U + Uc));
temp2 = x2 * (1 - c1 / R1) + c1 * U + c1 * Uc + c1 * x3 / R1;
temp3 = x3 * c2 + c3 * x2;
sub1 = Flatten[Flatten[Solve[temp1 == x1p, x1]] /. dAd];
sub2 = Flatten[Flatten[Solve[temp2 == x2p, x2]] /. dAd];
sub3 = Flatten[Flatten[Solve[temp3 == x3p, x3]] /. dAd];
temp4 = Flatten[sub2 /. sub3];
temp5 = Flatten[x2m /. temp4];
temp6 = Flatten[Solve[temp5 == x2m, x2m]];
temp7 = x1m /. sub1;
temp8 = MatrixForm[Flatten[temp7 /. temp6]]
```

$$\frac{c1\ Tc\ Uc + c1\ Th\ Um - \frac{x1\ (c1\ c2\ R1\ Uc+c1\ c2\ R1\ Um-c2\ R1\ x2+c1\ x3)}{c1\ c2+c1\ c3-c2\ R1}}{c1\ Uc + c1\ Um - \frac{c1\ c2\ R1\ Uc+c1\ c2\ R1\ Um-c2\ R1\ x2+c1\ x3}{c1\ c2+c1\ c3-c2\ R1}}$$

```
c = c1 c2 + c1 c3 - c2 R1;
tempx3h = (c1 c2 R1 Uc - c2 R1 x2 + c1 x3) / c;
temp9 = Flatten[(c1 Tc Uc + c1 Th Um - x1 * c1 c2 R1 Um / c - x1 * x3h) / c1 Uc + c1 Um - c1 c2 R1 Um / c - x3h /. Ad];
temp10 = Flatten[Solve[temp9 == x1, x3hp]];
temp11 = Flatten[x3hp /. temp10];
f1[x_, y_, z_] = MatrixForm[FullSimplify[Flatten[temp11 /. {U → x, x1 → y, x1p → z}]]]
temp12 = Flatten[Solve[tempx3h == x3h, x3]];
temp13 = Flatten[Solve[temp1 == x1p, x2]];
temp14 = temp2 /. temp12 /. temp13;
f2[x_, y_, z_, w_] =
 MatrixForm[FullSimplify[Flatten[temp14 /. {U → x, x1 → y, x1p → z, x3h → w}]]]
f3[x_, y_, z_, w_] = MatrixForm[FullSimplify[Flatten[temp1 /. {U → x, x1 → y, x2 → z}]]]
```

$$-\frac{c1\ c2\ R1\ x}{c1\ (c2+c3) - c2\ R1} + \frac{c1\ (-Tc\ Uc - Th\ x + (Uc + x)\ y)}{y - z}$$

$$\frac{1}{R1\ (y-z)}\Big(c1^2\ (Tc\ Uc + Th\ x - (Uc + x)\ y) + c2\ R1\ w\ (-y + z) - c1\ (-(c2 + c3)\ w\ (y - z) + R1\ ((1 + c2)\ Tc\ Uc + Th\ x + (Uc + x)\ (-2\ y + z) + c2\ (Th\ x - x\ y - Uc\ z)))\Big)$$

$$y + \frac{c1\ (Tc\ Uc + Th\ x - (Uc + x)\ y)}{z}$$

# REFERENCES


[1] J. Boothby, *An Introduction to Differentiable Manifolds and Riemanian Geometry*, 2nd ed., Orlando, FL: Academic, 1986.

[2] O. Bretscher, *Linear Algebra With Applications*, 3rd ed., Upper Saddle River, NJ: Pearson Prentice Hall, 2005.

[3] B. Jakubczyk, "Invertible realizations of nonlinear discrete-time systems," in Proc. Conf. Information Sciences Systems, 1980.

[4] Ü. Kotta, P. Liu and A. Zinober, "Transfer Equivalence and Realization of Nonlinear Higher Order Input-Output Difference Equations," *Automatica*, vol 37, pp. 1771-1778, 2001.

[5] Ü. Kotta and N. Sadegh, "Two Approaches for State Space Realization of NARMA Models: Bridging the Gap," Mathematical and Computer Modelling of Dynamical Systems, vol 8, no. 1, pp. 21-32, 2002.

[6] I. Leontaritis and S. Billings, "Input–output parametric models for nonlinear systems. part i: Deterministic nonlinear systems," Int. J. Control, vol. 41, no. 4, pp. 303–328, 1985.

[7] I. Leontaritis and S. Billings, "Input–output parametric models for nonlinear systems. part ii: Stochastic nonlinear systems," Int. J. Control, vol. 41, no. 4, pp. 329–344, 1985.

[8] A. Levin and K. Narendra, "Control of nonlinear dynamical systems using neural networks-controllability and stabilization," IEEE Trans. Neural Networks, vol. 4, pp. 192–206, Apr. 1993.

[9] L. Ljung, *System Identification: Theory for the User*, 2nd ed., Upper Saddle River, NJ: Prentice Hall,1999.

[10] S. Monaco and D. Norman-Cyrot, "On the realization of nonlinear discrete-time systems," Syst. Control Lett., vol. 2, pp. 145–152, 1984.

[11] S. Monaco and D. Norman-Cyrot, "Finite volterra series realizations and input–output approximation of nonlinear discrete-time systems," Int. J. Control, vol. 47, pp. 1771–1787, 1987.

[12] N. Sadegh, "Minimal Realization of Nonlinear Systems Described by Input-Output Difference Equations," IEEE Transactions on Automatic Control, vol 46, no. 5, pp. 698–710, May 2001.

[13] J.-J. Slotine, *Applied Nonlinear Control*. Englewood Cliffs, NJ: Prentice Hall, 1991.

[14] E. Sontag, Polynomial Response Maps. New York: Springer-Verlag, 1979.

[15] F. Szidarovszky and A. T. Bahill, *Linear Systems Theory*, 2nd ed., Boca Raton, Fl: CRC Press LLC, 1998.

[16] A. Van der Schaft, "On realizations of nonlinear systems described by higher-order differential equations," Math Systems Theory, vol. 19, pp. 239–275, 1987.

[17] M. Vidyasagar, *Nonlinear Systems Analysis*, 2nd ed., Philadelphia, PA: SIAM, 2002.

[18] Y. Wang and E. Sontag, "Algebraic differential equations and rational control systems," SIAM J. Control Optim., vol. 30, pp. 1126–1149, 1992.

[19] Weisstein, Eric W. "Nonlinear Least Squares Fitting." From MathWorld–A Wolfram Web Resource. http://mathworld.wolfram.com/NonlinearLeastSquaresFitting.html

[20] K. Ogata, Discrete-Time Control Systems, 2nd ed., Upper Saddle River, NJ: Prentice-Hall, 1994.